

[illegible]


```
1 0001 0 MODULE COB$DISPLAY ( %TITLE 'VAX-11 COBOL DISPLAY statement'
2 0002 0 IDENT = '1-015' ) = ! File: COBDISPLA.B32 EDIT:LGB1015
3 0003 0
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1
31 0031 1 ++
32 0032 1 FACILITY: COBOL SUPPORT
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1
36 0036 1 Supports the COBOL DISPLAY and DISPLAY WITH NO ADVANCING
37 0037 1 statements. Enhanced to perform the new screen handling
38 0038 1 extensions for VAX-11 COBOL Version 3.
39 0039 1
40 0040 1 Contains COB$$OPEN_OUT to open an RMS file for output.
41 0041 1
42 0042 1 Avoids use of STR$CONCAT to avoid its overhead. The
43 0043 1 concatenation which needs to be done is done inline since
44 0044 1 all required lengths are known.
45 0045 1
46 0046 1 ENVIRONMENT: VAX-11 User Mode
47 0047 1
48 0048 1 AUTHOR: Rich Reichert, CREATION DATE: 17-JULY-79
49 0049 1
50 0050 1 MODIFIED BY:
51 0051 1
52 0052 1 1-001 - Original. RKR 17-JULY-79
53 0053 1 1-002 - Remove usage of STR$CONCAT and associated string routines,
54 0054 1 as well as minor code rearrangements to improve resulting
55 0055 1 code. RKR 17-SEPT-79
56 0056 1 1-003 - Change basic algorithm for concatenation.
57 0057 1 If 1 string, write from caller's buffer
```

```
58 0058 1 | If more than 1 string and total length less than 132 chars,  
59 0059 1 | concatenated on stack.  
60 0060 1 | Else concatenate in heap storage.  
61 0061 1 | RKR 25-SEPT-79  
62 0062 1 | 1-004 - Change symbolic name of LIBRARY file. RKR 1-OCT-79  
63 0063 1 | 1-005 - Change references to LIB$ INVARG to COB$ INVARG  
64 0064 1 | Cosmetic changes. RKR 21-OCT-79  
65 0065 1 | 1-006 - Make sensitive to REQUIRE file. RKR 21-OCT-79  
66 0066 1 | 1-007 - Improve error messages. RKR 21-OCT-79  
67 0067 1 | 1-008 - Pass filename descriptor to COB$$WRITE RMS so that we have  
68 0068 1 | filename available for signaling if errors arise.  
69 0069 1 | RKR 05-NOV-79  
70 0070 1 | 1-009 - Make smaller by creating additional common code.  
71 0071 1 | RKR 07-NOV-79  
72 0072 1 | 1-010 - Imperative clean-ups, also try SYSS logicals.  
73 0073 1 | PDG 00-FEB-81  
74 0074 1 | 1-011 - Added EDIT phrase so CHECKIN creates a valid audit trail. Also  
75 0075 1 | updated copyright date. LB 9-AUG-81  
76 0076 1 | 1-012 - Added routines to perform the new screen handling extensions for  
77 0077 1 | Version 3 of VAX-11 COBOL.  
78 0078 1 | New routines;  
79 0079 1 | COB$DISP_SCR  
80 0080 1 | COB$DISP_SCR NO_ADV  
81 0081 1 | COMMON_SCREEN  
82 0082 1 | DISP_CONVERT  
83 0083 1 | DISP_PARSE  
84 0084 1 | COB$$FREE_STRINGS  
85 0085 1 | Changes to old routines;  
86 0086 1 | COB$$OPEN_OUT - made it a GLOBAL routine  
87 0087 1 | LGB 11-MAR-83  
88 0088 1 | 1-013 - Additional code for screen handling extensions.  
89 0089 1 | ADJUST_FL_PT macro.  
90 0090 1 | Version 1 / Version 3 DISPLAY and ACCEPT statement interaction.  
91 0091 1 | COB$$AB PREV initial state changed from 0 to 9.  
92 0092 1 | VAX-11 COBOL Compiler now passing a bit in FLAGS if they do  
93 0093 1 | not want a SIGN printed in a COMP data item.  
94 0094 1 | PIC P data item addressed.  
95 0095 1 | LGB 15-AUG-83  
96 0096 1 | 1-014 - Added routine to return address of COB$$AB PREV. This is needed  
97 0097 1 | by RPG$DSPLY. Routine is COB$$RET_A AB PREV. MDL 29-AUG-1983  
98 0098 1 | 1-015 - Parameter added to COB$$OPEN_OUT to bypass some COBOL code for  
99 0099 1 | RPG. Now uses COBPROLOG.REQ. LGB 24-OCT-83  
100 0100 1 | --
```



```
102 0101 1 |
103 0102 1 | PROLOGUE FILE
104 0103 1 |
105 0104 1 | REQUIRE 'RTLIN:COBPROLOG' ;
106 1621 1 |                               | Switches, Psects,
107 1622 1 | LINKAGES:                     | Include files.
108 1623 1 |
109 1624 1 | LINKAGE
110 1625 1 |     CVT_JSB = JSB (REGISTER = 6, REGISTER = 7, REGISTER = 8, REGISTER = 9) :
111 1626 1 |                     NOPRESERVE (2, 3, 4, 5, 6, 7, 8)
112 1627 1 |                     NOTUSED (10, 11) ;
113 1628 1 |
114 1629 1 | TABLE OF CONTENTS:
115 1630 1 |
116 1631 1 | FORWARD ROUTINE
117 1632 1 |     COB$DISPLAY:              NOVALUE,      | Display with normal advancing
118 1633 1 |     COB$DISP_NO_ADV:         NOVALUE,      | Display with no advancing
119 1634 1 |     COB$DISP_SCR:           NOVALUE,      | Display with screen enhancements
120 1635 1 |                               | and normal advancing
121 1636 1 |     COB$DISP_SCR_NO_ADV:     NOVALUE,      | Display with screen enhancements
122 1637 1 |                               | and no advancing
123 1638 1 |     COMMON_CODE:             NOVALUE,      | Code which is common to
124 1639 1 |                               | COB$DISPLAY and COB$DISP_NO_ADV
125 1640 1 |     COMMON_CODE_1:           NOVALUE,      | Code which is common to
126 1641 1 |                               | COB$DISPLAY and COB$DISP_NO_ADV
127 1642 1 |     COB$$OPEN_OUT:           NOVALUE,      | Open for output
128 1643 1 |     COMMON_SCREEN:           NOVALUE,      | Code which is common to COB$DISP_SCR
129 1644 1 |                               | and COB$DISP_SCR_NO_ADV
130 1645 1 |     DISP_CONVERT:            NOVALUE,      | Numeric conversions
131 1646 1 |     DISP_PARSE:              NOVALUE,      | Put together string for output
132 1647 1 |     COB$$FREE_STRINGS,       | Free local strings
133 1648 1 |     COB$$RET_A_AB_PREV ;     | Retrn address of COB$$AB_PREV
134 1649 1 |
135 1650 1 | EQUATED SYMBOLS:
136 1651 1 |
137 1652 1 | LITERAL
138 1653 1 |     NUM_UNITS = COB$K_UNIT_MAX - COB$K_UNIT_MIN + 1 ;    ! Number of units
139 1654 1 |
140 1655 1 | LITERAL
141 1656 1 |     DISP      = 0,      | Code for DISPLAY
142 1657 1 |     DNA       = 1,      | Code for DISPLAY with no advancing
143 1658 1 |     POS       = 2,      | Code for COB$POS_ERASE being called prior to
144 1659 1 |                               | entrance to this module (prev - display)
145 1660 1 |     POS_DNA   = 3,      | Code for COB$POS_ERASE being called prior to
146 1661 1 |                               | entrance to this module (prev - disp no adv)
147 1662 1 |     ACC_ADV   = 4,      | Code for ACCEPT Advancing (V3)
148 1663 1 |     ACC_DNA   = 5,      | Code for ACCEPT No Advancing (V3)
149 1664 1 |     CRR       = %X'8D',  | Code for carriage return
150 1665 1 |     LINE_FEED = %X'8A',  | Code for line-feed
151 1666 1 |     V_BELL    = 16,      | Bit flag for terminal bell
152 1667 1 |     V_CONV    = 32,      | Bit flag for conversion
153 1668 1 |     V_DEC_PT  = 64,      | Bit flag for 'decimal point is comma'
154 1669 1 |     V_NO_SIGN = 128,     | Bit flag for COMP data items, 1=do not print sign
155 1670 1 |     V_COB_RPG = 2048,    | Bit flag for VAX COBOL / VAX RPG
156 1671 1 |     FLAG_MASK = 15 ;     | Masks first four bits of FLAGS (0-3) for call
157 1672 1 |                               | to COB$$SET_ATTRIBUTES (bold, reverse, blink,
158 1673 1 |                               | and underline)
```



```
159 1674 1 |
160 1675 1 | GUARDS:
161 1676 1 |
162 1677 1 | Since the code assumes that COB$K_UNIT_MIN equals 0, and COB_TABLE
163 1678 1 | has only 7 items in it, we safeguard this module.
164 1679 1 |
165 1680 1 | %IF COB$K_UNIT_MIN NEQ 0 %THEN %ERROR('Unexpected COB$K_UNIT_MIN value') %FI
166 1681 1 | %IF COB$K_UNIT_MAX GTR 6 %THEN %ERROR('Unexpected COB$K_UNIT_MAX value') %FI
167 1682 1 |
168 1683 1 |
169 1684 1 | OWN STORAGE:
170 1685 1 |
171 1686 1 | The following GLOBAL cells are used by the file I/O routines.
172 1687 1 |
173 1688 1 | GLOBAL
174 1689 1 | COB$$AL_WRITE_RAB: VECTOR[ NUM UNITS ]
175 1690 1 | INITIAL (REP NUM UNITS OF LONG (0)), ! Address of output RAB
176 1691 1 | COB$$AW_WRITE_IFI: VECTOR[ NUM UNITS, WORD ]
177 1692 1 | INITIAL (REP NUM UNITS OF WORD (0)), ! Internal file identifiers
178 1693 1 | COB$$AB_USPCODE: VECTOR[ 2, BYTE ], ! byte 0 is prefix upspacing
179 1694 1 | ! byte 1 is post upspacing
180 1695 1 | COB$$AB_PREV: VECTOR[ NUM UNITS, BYTE ]
181 1696 1 | INITIAL (REP NUM UNITS OF BYTE (9)) ; ! History of whether previous call was
182 1697 1 | ! a DISPLAY or DISPLAY_NO_ADV
183 1698 1 |
184 1699 1 | MACROS:
185 1700 1 | Adjust the output of the OTSS routines that convert Floating Point
186 1701 1 | and Double Floating Point to Text.
187 1702 1 | Old result 0.1110000E+03, now want 1.110000E+02
188 1703 1 |
189 1704 1 | MACRO
190 M 1705 1 | ADJUST_FL_PT =
191 M 1706 1 |
192 M 1707 1 | BEGIN ! Begin FL macro
193 M 1708 1 |
194 M 1709 1 | LOCAL
195 M 1710 1 | ANS_BUF : REF VECTOR [25,BYTE],
196 M 1711 1 | E_SIGN, ! sign position of exp
197 M 1712 1 | E_ONES, ! ones position of exp
198 M 1713 1 | E_TENS, ! tens position of exp
199 M 1714 1 | SHIFT_ALL, ! # of chars to shift
200 M 1715 1 | SEARCH, ! # of chars to search
201 M 1716 1 | CHANGE, ! =1 if e+00 should be
202 M 1717 1 | ! changed to e-01
203 M 1718 1 | TEMP : BYTE ; ! Needed for exchange
204 M 1719 1 |
205 M 1720 1 | ANS_BUF = .ANS_STRING [DSC$A_POINTER] ;
206 M 1721 1 | IF T.STRING [DSC$B_DTYPE] EQ [DSC$K_DTYPE_F ]
207 M 1722 1 | THEN
208 M 1723 1 | BEGIN ! Floating
209 M 1724 1 | E_SIGN = 11 ; ! Where to find exponent
210 M 1725 1 | E_ONES = 13 ; ! in ANS_STRING
211 M 1726 1 | E_TENS = 12 ;
212 M 1727 1 | SHIFT_ALL = 12 ; ! Shift 12 chars for FL
213 M 1728 1 | END
214 M 1729 1 | ELSE ! Double Floating
215 M 1730 1 | BEGIN
```



```
216 M 1731 1 E_SIGN = 20 ; ! Where to find exponent
217 M 1732 1 E_ONES = 22 ; ! in ANS_STRING
218 M 1733 1 E_TENS = 21 ;
219 M 1734 1 SHIFT_ALL = 21 ; ! Shift 21 chars for D FL
220 M 1735 1 END ;
221 M 1736 1
222 M 1737 1
223 M 1738 1 !+ 'Decimal Point is Comma' - place comma in ANS_BUF to overwrite
224 M 1739 1 !- decimal point that is already there.
225 M 1740 1
226 M 1741 1
227 M 1742 1 IF (.FLAGS AND V_DEC_PT) NEQ 0
228 M 1743 1 THEN
229 M 1744 1 ANS_BUF [2] = %C',' ;
230 M 1745 1
231 M 1746 1 !+
232 M 1747 1 !- Adjust exponent - decrement if positive, increment if negative.
233 M 1748 1 !- However leave 0.00..00E+00 as is.
234 M 1749 1
235 M 1750 1
236 M 1751 1 IF (.ANS_BUF [.E_ONES] EQL %C'0' ) AND (.ANS_BUF [.E_TENS] EQL %C'0')
237 M 1752 1 THEN
238 M 1753 1 BEGIN
239 M 1754 1 !+
240 M 1755 1 !- E+00 -> E-01 -> but leave 0.00...00E+00 as is.
241 M 1756 1
242 M 1757 1 IF .SHIFT_ALL EQL 12
243 M 1758 1 THEN SEARCH = 9
244 M 1759 1 ELSE SEARCH = 18 ;
245 M 1760 1 INCR P FROM 3 TO .SEARCH DO
246 M 1761 1 IF .ANS_BUF [.P] NEQ %C'0'
247 M 1762 1 THEN CHANGE = 1 ;
248 M 1763 1
249 M 1764 1 IF .CHANGE EQL 1 ! Change xxxE+00 to
250 M 1765 1 THEN ! xxxE-01
251 M 1766 1 BEGIN
252 M 1767 1 ANS_BUF [.E_SIGN] = %C'-' ;
253 M 1768 1 ANS_BUF [.E_TENS] = %X'30' ;
254 M 1769 1 ANS_BUF [.E_ONES] = %X'31' ;
255 M 1770 1 END ;
256 M 1771 1 END
257 M 1772 1 ELSE
258 M 1773 1 BEGIN
259 M 1774 1 IF (.ANS_BUF [.E_SIGN] EQL %C'+' )
260 M 1775 1 THEN
261 M 1776 1 !+
262 M 1777 1 !- Exponent is positive - decrement it
263 M 1778 1
264 M 1779 1 BEGIN
265 M 1780 1 IF (.ANS_BUF [.E_ONES] NEQ %C'0')
266 M 1781 1 THEN
267 M 1782 1 ANS_BUF [.E_ONES] = .ANS_BUF [.E_ONES] - 1 ! E+18 -> E+17
268 M 1783 1 ELSE
269 M 1784 1 BEGIN ! E+20 -> E+19
270 M 1785 1 ANS_BUF [.E_ONES] = %X'39' ;
271 M 1786 1 ANS_BUF [.E_TENS] = .ANS_BUF [.E_TENS] - 1 ;
272 M 1787 1 END ;
```



```

: 273 M 1788 1      END
: 274 M 1789 1      ELSE
: 275 M 1790 1      | +
: 276 M 1791 1      | Exponent negative - increment it
: 277 M 1792 1      | -
: 278 M 1793 1      BEGIN
: 279 M 1794 1      IF (.ANS_BUF [.E_ONES] NEQ %C'9')
: 280 M 1795 1      THEN
: 281 M 1796 1          ANS_BUF [.E_ONES] = .ANS_BUF [.E_ONES] + 1      ! E-15 -> E-16
: 282 M 1797 1      ELSE
: 283 M 1798 1          BEGIN
: 284 M 1799 1              ANS_BUF [.E_ONES] = %X'30' ;
: 285 M 1800 1              ANS_BUF [.E_TENS] = .ANS_BUF [.E_TENS] + 1 ;
: 286 M 1801 1          END ;
: 287 M 1802 1      END ;
: 288 M 1803 1      END ;
: 289 M 1804 1
: 290 M 1805 1      | +
: 291 M 1806 1      | Exchange decimal point and digit you want before the decimal point
: 292 M 1807 1      | Mantissa was 0.123... now will be 01.23...
: 293 M 1808 1      | -
: 294 M 1809 1
: 295 M 1810 1      TEMP = .ANS_BUF [2] ;
: 296 M 1811 1      ANS_BUF [2] = .ANS_BUF [3] ;
: 297 M 1812 1      ANS_BUF [3] = .TEMP ;
: 298 M 1813 1
: 299 M 1814 1      | +
: 300 M 1815 1      | Pull the zero that is before the decimal point. Shift the other
: 301 M 1816 1      | digits by one.
: 302 M 1817 1          Old result      0.1110000E+03,
: 303 M 1818 1          from exchange above 01.110000E+02,
: 304 M 1819 1          now want      1.110000E+02
: 305 M 1820 1      | -
: 306 M 1821 1
: 307 M 1822 1      INCR X FROM 1 TO .SHIFT_ALL DO
: 308 M 1823 1          ANS_BUF [.X] = .ANS_BUF [.X+1] ;
: 309 M 1824 1
: 310 M 1825 1      | +
: 311 M 1826 1      | Adjust length of ANS_STRING
: 312 M 1827 1      | -
: 313 M 1828 1
: 314 M 1829 1      ANS_STRING [DSC$W_LENGTH] = .ANS_STRING [DSC$W_LENGTH] - 1 ;
: 315 M 1830 1
: 316 M 1831 1      END ;
: 317 M 1832 1      % ;
: 318 M 1833 1
: 319 M 1834 1      |
: 320 M 1835 1      | The following tables convert the UNIT number into a logical name.
: 321 M 1836 1      |
: 322 M 1837 1      MACRO
: 323 M 1838 1          DESC_(A) = UPLIT BYTE(%ASCIC A) - BASE %;
: 324 M 1839 1      BIND
: 325 M 1840 1          BASE = UPLIT(REP 0 OF (0)),
: 326 M 1841 1          COB_TABLE = UPLIT(
: 327 M 1842 1              DESC_('COB$INPUT'),
: 328 M 1843 1              DESC_('COB$OUTPUT'),
: 329 M 1844 1              DESC_('COB$CONSOLE'),

```



```
: 330      1845 1      DESC-('COB$CARDREADER'),
: 331      1846 1      DESC-('COB$PAPERTAPEREREADER'),
: 332      1847 1      DESC-('COB$LINEPRINTER'),
: 333      1848 1      DESC-('COB$PAPERTAPEPUNCH')): VECTOR[UM_UNITS],
: 334      1849 1      SYS_TABLE = UPLIT(
: 335      1850 1      DESC-('SY$$INPUT'),
: 336      1851 1      DESC-('SY$$OUTPUT'),
: 337      1852 1      DESC-('SY$$ERROR'),
: 338      1853 1      DESC-('SY$$INPUT'),
: 339      1854 1      DESC-('SY$$INPUT'),
: 340      1855 1      DESC-('SY$$OUTPUT'),
: 341      1856 1      DESC-('SY$$OUTPUT')): VECTOR[UM_UNITS];
: 342      1857 1
: 343      1858 1
: 344      1859 1      ! EXTERNAL REFERENCES:
: 345      1860 1
: 346      1861 1      EXTERNAL ROUTINE
: 347      1862 1      LIB$STOP : NOVALUE,
: 348      1863 1      LIB$GET_VM,
: 349      1864 1      LIB$FREE_VM,
: 350      1865 1      STR$DUPL_CHAR,
: 351      1866 1      STR$GET1_DX,
: 352      1867 1      STR$FREE1_DX,
: 353      1868 1      STR$COPY_R,
: 354      1869 1      COB$CNVOUT,
: 355      1870 1
: 356      1871 1      COB$CVTQP R9 : CVT JSB,
: 357      1872 1      COB$$SETUP_TERM_TYPE,
: 358      1873 1      COB$$SET_ATTRIBUTES ;
: 359      1874 1
: 360      1875 1      EXTERNAL LITERAL
: 361      1876 1      COB$_ERRDURDIS,
: 362      1877 1      COB$_FAIGET_VM,
: 363      1878 1      COB$_INVARG;
: 364      1879 1
: 365      1880 1      EXTERNAL
: 366      1881 1      COB$TERM_TYPE;

! Signals fatal error
! Get virtual memory
! Free virtual memory
! Duplicate a character n times
! Allocate a string
! Deallocate a string
! Copy a string by ref
! Convert from D and F floating
! to Fortran E format
! Convert quad to packed
! Setup terminal type
! Set bold, reverse, blink,
! underline

! Error during DISPLAY
! Failure to get VM
! Invalid Argument(s)

! Terminal type
```



```

368 1882 1 GLOBAL ROUTINE COB$DISPLAY (
369 1883 1     UNIT,          ! Unit # of output device
370 1884 1     STRING      ! Input string
371 1885 1 ) : NOVALUE =
372 1886 1 ++
373 1887 1
374 1888 1 FUNCTIONAL DESCRIPTION:
375 1889 1
376 1890 1     Performs COBOL DISPLAY statement given a unit number and
377 1891 1     one or more strings to display.  If more than one string is
378 1892 1     specified, these strings are concatenated into a single string
379 1893 1     before being output.  The upspacing to be employed is a function
380 1894 1     of this call (normal ADVANCING) and the upspacing used on a
381 1895 1     previous call to this routine or to COB$DISP_NO_ADV, or
382 1896 1     COB$DISP_SCR, or COB$DISP_SCR_NO_ADV.
383 1897 1
384 1898 1
385 1899 1 FORMAL PARAMETERS:
386 1900 1
387 1901 1     UNIT.rl.v      integer unit number designating the device
388 1902 1                  on which the string(s) is(are) to be displayed.
389 1903 1
390 1904 1     STRING.rt.dx   address of 1st of up to 254 string descriptors
391 1905 1                  which are to concatenated and displayed on the
392 1906 1                  specified device.
393 1907 1
394 1908 1 IMPLICIT INPUTS:
395 1909 1
396 1910 1     Status information as to whether the output file in question
397 1911 1     is currently open.
398 1912 1
399 1913 1 IMPLICIT OUTPUTS:
400 1914 1
401 1915 1     Updated status information for this file.
402 1916 1
403 1917 1 ROUTINE VALUE:
404 1918 1
405 1919 1     NONE
406 1920 1
407 1921 1 SIDE EFFECTS:
408 1922 1
409 1923 1     Outputs a record on the specified file.
410 1924 1 --
411 1925 2 BEGIN
412 1926 2 BUILTIN
413 1927 2 CALLG,
414 1928 2 AP;
415 1929 2
416 1930 2 COB$$AB USPCODE[1] = CRR;          ! Upspace code is carriage return
417 1931 2 CALLG(.AP, COMMON_CODE_1);
418 1932 2 COB$$AB_PREV[0] = "DISP";      ! Prev. unit to become DISPLAY
419 1933 2
420 1934 1 END;
```



```
.PSECT _COB$DATA,NOEXE, PIC,2

00000000# 00000 COB$$AL_WRITE_RAB::
               .LONG 0[7]
0000# 0001C COB$$AW_WRITE_IFI::
               .WORD 0[7]
               0002A .BLKB 2
               0002C COB$$AB_USPCODE::
               .BLKB 2
               0002E .BLKB 2
09# 00030 COB$$AB_PREV::
               .BYTE 9[7]

.PSECT _COB$CODE,NOWRT, SHR, PIC,2

52 45 44 41 45 54 55 50 4E 49 24 42 4F 43 09 00000 P.AAA: .BLKB 0
52 45 50 41 54 55 50 54 55 4F 24 42 4F 43 0A 00000 P.AAC: .ASCII <9>\COB$INPUT\
45 54 4E 49 52 50 45 4E 49 4C 24 42 4F 43 0B 0000A P.AAD: .ASCII <10>\COB$OUTPUT\
50 45 50 41 54 52 45 50 41 50 24 42 4F 43 0E 00015 P.AAE: .ASCII <11>\COB$CONSOLE\
45 54 4E 49 52 50 45 4E 49 4C 24 42 4F 43 0F 00021 P.AAF: .ASCII <14>\COB$CARDREADER\
50 45 50 41 54 52 45 50 41 50 24 42 4F 43 13 00030 P.AAG: .ASCII <19>\COB$PAPERTAPEREREADER\
45 54 4E 49 52 50 45 4E 49 4C 24 42 4F 43 0F 0003F P.AAH: .ASCII <15>\COB$LINEPRINTER\
50 45 50 41 54 52 45 50 41 50 24 42 4F 43 12 00053 P.AAI: .ASCII <18>\COB$PAPERTAPEPUNCH\
45 54 4E 49 52 50 45 4E 49 4C 24 42 4F 43 12 00063 P.AAB: .BLKB 1
50 45 50 41 54 52 45 50 41 50 24 42 4F 43 12 00067 P.AAB: .LONG 0, 10, 21, 33, 48, 68, 84
45 54 4E 49 52 50 45 4E 49 4C 24 42 4F 43 12 00080 P.AAK: .ASCII <9>\SYSS$INPUT\
50 45 50 41 54 52 45 50 41 50 24 42 4F 43 12 00084 P.AAL: .ASCII <10>\SYSS$OUTPUT\
45 54 4E 49 52 50 45 4E 49 4C 24 42 4F 43 12 0008E P.AAM: .ASCII <9>\SYSS$ERROR\
50 45 50 41 54 52 45 50 41 50 24 42 4F 43 12 00099 P.AAN: .ASCII <9>\SYSS$INPUT\
45 54 4E 49 52 50 45 4E 49 4C 24 42 4F 43 12 000AD P.AAO: .ASCII <9>\SYSS$INPUT\
50 45 50 41 54 52 45 50 41 50 24 42 4F 43 12 000B7 P.AAP: .ASCII <10>\SYSS$OUTPUT\
45 54 4E 49 52 50 45 4E 49 4C 24 42 4F 43 12 000C2 P.AAQ: .ASCII <10>\SYSS$OUTPUT\
50 45 50 41 54 52 45 50 41 50 24 42 4F 43 12 000CD P.AAJ: .BLKB 3
45 54 4E 49 52 50 45 4E 49 4C 24 42 4F 43 12 000D0 P.AAJ: .LONG 132, 142, 153, 163, 173, 183, 194
50 45 50 41 54 52 45 50 41 50 24 42 4F 43 12 000E8

BASE=
COB_TABLE=
SYS_TABLE=
P.AAA
P.AAB
P.AAJ
LIB$STOP, LIB$GET_VM
LIB$FREE_VM, STR$DUPL_CHAR
STR$GET1_DX, STR$FREET_DX
STR$COPY_R, COB$CNVOUT
COB$CVTQP R9, COB$$SETUP_TERM_TYPE
COB$$SET_ATTRIBUTES
COB$ERRDURDIS, COB$ FAIGET_VM
COB$_INVARG, COB$TERM_TYPE

00000000' EF 8D 8F 90 00002
0000V CF 6C FA 0000A
00000000' EF 94 0000F

.ENTRY COB$DISPLAY, Save nothing
MOVW #-115, COB$$AB_USPCODE+1
CALLG (AP), COMMON_CODE_1
CLRB COB$$AB_PREV
```

```
: 1882
: 1930
: 1931
: 1932
```

COB\$DISPLAY
1-015

VAX-11 COBOL DISPLAY statement

E 4
16-Sep-1984 00:02:31
14-Sep-1984 12:10:42

VAX-11 Bliss-32 V4.0-742
[COBRTL.SRC]COBDISPLA.B32;1

Page 10
(3)

04 00015

RET

; 1934

; Routine Size: 22 bytes, Routine Base: _COB\$CODE + 00EC


```

422 1935 1 GLOBAL ROUTINE COB$DISP_NO_ADV (
423 1936 1     UNIT,                                ! Unit # of output device
424 1937 1     STRING                            ! Input string
425 1938 1     ) : NOVALUE =
426 1939 1
427 1940 1 ++
428 1941 1 FUNCTIONAL DESCRIPTION:
429 1942 1
430 1943 1     Performs COBOL DISPLAY with NO ADVANCING statement given a unit number and
431 1944 1     one or more strings to display. If more than one string is
432 1945 1     specified, these strings are concatenated into a single string
433 1946 1     before being output. The upspacing to be employed is a function
434 1947 1     of this call (NO ADVANCING) and the upspacing used on a
435 1948 1     previous call to this routine or to COB$DISPLAY or COB$DISP_SCR,
436 1949 1     or COB$DISP_SCR_NO_ADV.
437 1950 1
438 1951 1
439 1952 1 FORMAL PARAMETERS:
440 1953 1
441 1954 1     UNIT.rl.v      integer unit number designating the device
442 1955 1                   on which the string(s) is(are) to be displayed.
443 1956 1
444 1957 1     STRING.rt.dx    address of 1st of up to 254 string descriptors
445 1958 1                   which are to concatenated and displayed on the
446 1959 1                   specified device.
447 1960 1
448 1961 1 IMPLICIT INPUTS:
449 1962 1
450 1963 1     Status information as to whether the output file in question
451 1964 1     is currently open.
452 1965 1
453 1966 1 IMPLICIT OUTPUTS:
454 1967 1
455 1968 1     Updated status information for this file.
456 1969 1
457 1970 1 ROUTINE VALUE:
458 1971 1
459 1972 1     NONE
460 1973 1
461 1974 1 SIDE EFFECTS:
462 1975 1
463 1976 1     Outputs a record on the specified file.
464 1977 1
465 1978 1 --
466 1979 2 BEGIN
467 1980 2 BUILTIN
468 1981 2 CALLG,
469 1982 2 AP;
470 1983 2
471 1984 2 COB$$AB_USPCODE[1] = 0;           ! Upspace code is 0
472 1985 2 CALLG(.AP, COMMON_CODE_1);
473 1986 2 COB$$AB_PREV[0] = DNA;          ! Prev. unit to become DISPLAY_NO_ADV
474 1987 2
475 1988 1 END;
```

COB\$DISPLAY
1-015

VAX-11 COBOL DISPLAY statement

G 4
16-Sep-1984 00:02:31
14-Sep-1984 12:10:42

VAX-11 Bliss-32 V4.0-742
[COBRTL.SRC]COBDISPLA.B32;1

Page 12
(4)

```
0000V CF 00000000' EF 0000 00000
00000000' EF 6C 94 00002
01 90 0000D
04 00014
```

```
.ENTRY COB$DISP NO ADV, Save nothing
CLRB COB$$AB_DSPCODE+1
CALLG (AP), COMMON CODE_1
MOVB #1, COB$$AB_PREV
RET
```

```
: 1935
: 1984
: 1985
: 1986
: 1988
```

; Routine Size: 21 bytes, Routine Base: _COB\$CODE + 0102


```

477 1989 1 GLOBAL ROUTINE COB$DISP_SCR (
478 1990 1
479 1991 1         UNIT,          ! Unit # of output device
480 1992 1         STRING,       ! Input string
481 1993 1         FLAGS        ! Screen enhancement flag
482 1994 1     ) : NOVALUE =
483 1995 1
484 1996 1     ++
485 1997 1     FUNCTIONAL DESCRIPTION:
486 1998 1
487 1999 1     Performs COBOL DISPLAY statement with screen enhancements.
488 2000 1     Given a unit number and one string to display using a flag that
489 2001 1     contains selected enhancements.
490 2002 1     A call to COB$POS_ERASE is made by the VAX-11 COBOL Compiler
491 2003 1     prior to the call to COB$DISP_SCR to set cursor position and
492 2004 1     perform any screen or line erasing.
493 2005 1     The upspacing to be employed is a function of COB$POS_ERASE and
494 2006 1     the upspacing used on a previous call to this routine or to
495 2007 1     either COB$DISPLAY, COB$DISP_NO_ADV or COB$DISP_SCR_NO_ADV.
496 2008 1
497 2009 1     FORMAL PARAMETERS:
498 2010 1
499 2011 1     UNIT.rl.v      integer unit number designating the device
500 2012 1                  on which the string is to be displayed.
501 2013 1
502 2014 1     STRING.rt.dx   address of string descriptor which is to be
503 2015 1                  displayed on the specified device.
504 2016 1
505 2017 1     FLAGS.rlu.v    screen enhancement flag;
506 2018 1
507 2019 1                  bit 0 - bold
508 2020 1                  bit 1 - reverse
509 2021 1                  bit 2 - blinking
510 2022 1                  bit 3 - underline
511 2023 1                  bit 4 - bell
512 2024 1                  bit 5 - conversion
513 2025 1                  bit 6 - decimal point is comma
514 2026 1                  bit 7 - 0 print sign, 1 do not print sign
515 2027 1                  bit 11 - 0 for VAX COBOL, 1 for VAX RPG
516 2028 1
517 2029 1
518 2030 1     IMPLICIT INPUTS:
519 2031 1
520 2032 1         Status information as to whether the output file in question
521 2033 1         is currently open.
522 2034 1
523 2035 1     IMPLICIT OUTPUTS:
524 2036 1
525 2037 1         Updated status information for this file.
526 2038 1
527 2039 1     ROUTINE VALUE:
528 2040 1
529 2041 1         NONE
530 2042 1
531 2043 1     SIDE EFFECTS:
532 2044 1
533 2045 1         Outputs a record on the specified file.
```

COB\$DISPLAY
1-015

VAX-11 COBOL DISPLAY statement

I 4
16-Sep-1984 00:02:31
14-Sep-1984 12:10:42

VAX-11 Bliss-32 V4.0-742
[COBRTL.SRC]COBDISPLA.B32;1

Page 14
(5)

```
: 534      2046  1  !--  
: 535      2047  2  BEGIN  
: 536      2048  2  BUILTIN  
: 537      2049  2  CALLG,  
: 538      2050  2  AP;  
: 539      2051  2  
: 540      2052  2  COB$$AB_USPCODE[1] = CRR;  
: 541      2053  2  CALLG(.AP, COMMON_SCREEN);  
: 542      2054  2  COB$$AB_PREV[0] = DISP;  
: 543      2055  2  
: 544      2056  1  END;
```

```
! Upspace code is carriage return  
! Do common processing  
! Prev. unit to become DISPLAY  
  
! end COB$DISP_SCR
```

```
00000000' EF      8D      8F 90 00002  
0000V CF      00000000' EF 94 0000F  
04 00015
```

```
.ENTRY COB$DISP_SCR, Save nothing  
MOVB #-115, COB$$AB_USPCODE+1  
CALLG (AP), COMMON_SCREEN  
CLRB COB$$AB_PREV  
RET
```

```
: 1989  
: 2052  
: 2053  
: 2054  
: 2056
```

; Routine Size: 22 bytes, Routine Base: _COB\$CODE + 0117


```
546 2057 1 GLOBAL ROUTINE COB$DISP_SCR_NO_ADV (
547 2058 1     UNIT,          ! Unit # of output device
548 2059 1     STRING,      ! Input string
549 2060 1     FLAGS       ! Screen enhancement flag
550 2061 1 ) : NOVALUE =
551 2062 1
552 2063 1 ++
553 2064 1 FUNCTIONAL DESCRIPTION:
554 2065 1
555 2066 1     Performs COBOL DISPLAY NO ADVANCING statement with screen
556 2067 1     enhancements. Given a unit number and one string to display using
557 2068 1     a flag that contains selected enhancements.
558 2069 1     A call to COB$POS_ERASE is made by the VAX-11 COBOL Compiler
559 2070 1     prior to the call to COB$DISP_SCR_NO_ADV to set cursor position
560 2071 1     and perform any screen or line erasing.
561 2072 1     The upspacing to be employed is a function of COB$POS_ERASE and
562 2073 1     the upspacing used on a previous call to this routine or to
563 2074 1     either COB$DISPLAY, COB$DISP_NO_ADV or COB$DISP_SCR.
564 2075 1
565 2076 1
566 2077 1 FORMAL PARAMETERS:
567 2078 1
568 2079 1     UNIT.rl.v      integer unit number designating the device
569 2080 1                  on which the string is to be displayed.
570 2081 1
571 2082 1     STRING.rt.dx   address of string descriptor which is to be
572 2083 1                  displayed on the specified device.
573 2084 1
574 2085 1     FLAGS.rlu.v    screen enhancement flag:
575 2086 1                  bit 0 - bold
576 2087 1                  bit 1 - reverse
577 2088 1                  bit 2 - blinking
578 2089 1                  bit 3 - underline
579 2090 1                  bit 4 - bell
580 2091 1                  bit 5 - conversion
581 2092 1                  bit 6 - decimal point is comma
582 2093 1                  bit 7 - 0 print sign, 1 do not print sign
583 2094 1                  bit 11 - 0 for VAX COBOL, 1 for VAX RPG
584 2095 1
585 2096 1
586 2097 1 IMPLICIT INPUTS:
587 2098 1
588 2099 1     Status information as to whether the output file in question
589 2100 1     is currently open.
590 2101 1
591 2102 1 IMPLICIT OUTPUTS:
592 2103 1
593 2104 1     Updated status information for this file.
594 2105 1
595 2106 1 ROUTINE VALUE:
596 2107 1
597 2108 1     NONE
598 2109 1
599 2110 1 SIDE EFFECTS:
600 2111 1
601 2112 1     Outputs a record on the specified file.
602 2113 1
```

COB\$DISPLAY
1-015

VAX-11 COBOL DISPLAY statement

K 4
16-Sep-1984 00:02:31
14-Sep-1984 12:10:42

VAX-11 Bliss-32 V4.0-742
[COBRTL.SRC]COBDISPLA.B32;1

Page 16
(6)

```
: 603      2114 1  !--  
: 604      2115 2  BEGIN  
: 605      2116 2  BUILTIN  
: 606      2117 2  CALLG,  
: 607      2118 2  AP;  
: 608      2119 2  
: 609      2120 2  COB$$AB_USPCODE[1] = 0;  
: 610      2121 2  CALLG(.AP, COMMON_SCREEN);  
: 611      2122 2  COB$$AB_PREV[0] = DNA;  
: 612      2123 2  
: 613      2124 1  END;
```

```
! Upspace code is 0  
! Do common processing  
! Prev unit to become DISPLAY_NO_ADV  
! end COB$DISP_SCR_NO_ADV
```

```
0000V CF 00000000' 0000 0000  
00000000' EF 94 00002  
6C FA 00008  
01 90 0000D  
04 00014
```

```
.ENTRY COB$DISP_SCR_NO_ADV, Save nothing  
CLRB COB$$AB_USPCODE+1  
CALLG (AP), COMMON_SCREEN  
MOVB #1, COB$$AB_PREV  
RET
```

```
: 2057  
: 2120  
: 2121  
: 2122  
: 2124
```

; Routine Size: 21 bytes, Routine Base: _COB\$CODE + 012D


```

615 2125 1 ROUTINE COMMON_CODE_1 (UNIT, STRING): NOVALUE =
616 2126 1 ++
617 2127 1
618 2128 1 FUNCTIONAL DESCRIPTION:
619 2129 1
620 2130 1     Performs common part of DISPLAY and DISPLAY_NO_ADV processing.
621 2131 1
622 2132 1
623 2133 1 FORMAL PARAMETERS:
624 2134 1
625 2135 1     UNIT.rl.v     integer unit number designating the device
626 2136 1                  on which the string(s) is(are) to be displayed.
627 2137 1
628 2138 1     STRING.rt.dx   address of 1st of up to 254 string descriptors
629 2139 1                  which are to concatenated and displayed on the
630 2140 1                  specified device.
631 2141 1
632 2142 1 IMPLICIT INPUTS:
633 2143 1
634 2144 1     Status information as to whether the output file in question
635 2145 1     is currently open.
636 2146 1
637 2147 1 IMPLICIT OUTPUTS:
638 2148 1
639 2149 1     Updated status information for this file.
640 2150 1
641 2151 1 ROUTINE VALUE:
642 2152 1
643 2153 1     NONE
644 2154 1
645 2155 1 SIDE EFFECTS:
646 2156 1
647 2157 1     Outputs a record on the specified file.
648 2158 1 --
649 2159 2 BEGIN
650 2160 2 BUILTIN
651 2161 2     ACTUALPARAMETER,
652 2162 2     ACTUALCOUNT;
653 2163 2
654 2164 2 LOCAL
655 2165 2     TEMP: VECTOR [COB$K_DIS_SIZE,BYTE], ! Temp buffer on stack
656 2166 2     COUNT, ! Total chars to output
657 2167 2     ADDR, ! Pointer into allocated storage
658 2168 2     STATUS, ! Status from LIB$GET_VM call
659 2169 2     DESC: BLOCK [8,BYTE]; ! Dynamically constructed desc.
660 2170 2 ! for concatenating strings
661 2171 2
662 2172 2
663 2173 2 ! If there is only one item to display, write directly from caller's buffer
664 2174 2
665 2175 2 IF ACTUALCOUNT() EQL 2
666 2176 2 THEN
667 2177 2 BEGIN
668 2178 2     COMMON_CODE(.UNIT, .STRING); ! Do common processing
669 2179 2     RETURN;
670 2180 2 END;
671 2181 2
```

```

672 2182 2
673 2183 2
674 2184 2
675 2185 2
676 2186 2
677 2187 2
678 2188 2
679 2189 2
680 2190 2
681 2191 2
682 2192 2
683 2193 2
684 2194 2
685 2195 2
686 2196 2
687 2197 2
688 2198 2
689 2199 2
690 2200 2
691 2201 2
692 2202 2
693 2203 2
694 2204 2
695 2205 2
696 2206 2
697 2207 2
698 2208 2
699 2209 2
700 2210 2
701 2211 2
702 2212 2
703 2213 2
704 2214 2
705 2215 2
706 2216 2
707 2217 2
708 2218 2
709 2219 2
710 2220 2
711 2221 2
712 2222 2
713 2223 2
714 2224 2
715 2225 2
716 2226 2
717 2227 2
718 2228 2
719 2229 2
720 2230 2
721 2231 2
722 2232 2
723 2233 2
724 2234 2
725 2235 2
726 2236 2
727 2237 2
728 2238 2

! Count total text to be displayed
COUNT = 0;
INCR I FROM 2 TO ACTUALCOUNT() DO
    BEGIN
        COUNT = .COUNT + .BLOCK[ACTUALPARAMETER(.I), DSC$W_LENGTH; , BYTE];
    END;

! Build a fixed string descriptor
DESC[DSC$W_LENGTH] = 0;
DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
DESC[DSC$A_POINTER] = TEMP; ! Assume stack is used

! Concatenate the caller's string(s) into a single string.
! If there are more than "COB$K_DIS_SIZE" characters to be displayed,
! allocate heap storage -- else use the stack.
IF .COUNT GTR COB$K_DIS_SIZE
THEN
    BEGIN
        ! Allocate space and store its address into descriptor
        IF NOT (STATUS = LIB$GET_VM(COUNT, DESC[DSC$A_POINTER]))
        THEN
            LIB$STOP(COB$FAIGET_VM, 0, .STATUS);
        END;

        ADDR = .DESC[DSC$A_POINTER];
        INCR I FROM 2 TO ACTUALCOUNT() DO
            BEGIN
                LOCAL
                PTR: REF BLOCK[BYTE];
                PTR = ACTUALPARAMETER(.I);
                CH$MOVE(.PTR[DSC$W_LENGTH], .PTR[DSC$A_POINTER], .ADDR);
                ADDR = .ADDR + .PTR[DSC$W_LENGTH];
            END;

        DESC[DSC$W_LENGTH] = .ADDR - .DESC[DSC$A_POINTER];

        COMMON_CODE(.UNIT, DESC); ! Do common processing

! If we've been using heap storage, give it back
IF .COUNT GTR COB$K_DIS_SIZE
THEN
    LIB$FREE_VM(COUNT, .DESC[DSC$A_POINTER]);
```


; 729 2239 1 END;

				03FC 00000 COMMON_CODE 1:			
		5E	FF70	CE 9E 00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9	2125
		02		6C 91 00007	MOVAB	-144(SP), SP	
				0A 12 0000A	CMPB	(AP), #2	2175
		7E	04	AC 7D 0000C	BNEQ	1\$	
	0000V	CF		02 FB 00010	MOVQ	UNIT, -(SP)	2178
					CALLS	#2, COMMON_CODE	
				04 00015	RET		2177
				6E D4 00016	CLRL	COUNT	2185
		52		6C 9A 00018	MOVZBL	(AP), R2	2186
		50		01 D0 0001B	MOVL	#1, I	
				0A 11 0001E	BRB	3\$	
		51		6C40 D0 00020	MOVL	(AP)[I], R1	2188
		53		61 3C 00024	MOVZWL	(R1), R3	
		6E		53 C0 00027	ADDL2	R3, COUNT	
F2		50		52 F3 0002A	AOBLEQ	R2, I, 2\$	2186
	04	AE	010E0000	8F D0 0002E	MOVL	#17694720, DESC	2194
	08	AE	0C	AE 9E 00036	MOVAB	TEMP, DESC+4	2197
	00000084	8F		6E D1 0003B	CMPL	COUNT, #132	2204
				21 15 00042	BLEQ	4\$	
			08	AE 9F 00044	PUSHAB	DESC+4	2210
			04	AE 9F 00047	PUSHAB	COUNT	
	00000000G	00		02 FB 0004A	CALLS	#2, LIB\$GET_VM	
		11		50 E8 00051	BLBS	STATUS, 4\$	
				50 DD 00054	PUSHL	STATUS	2212
				7E D4 00056	CLRL	-(SP)	
			00000000G	8F DD 00058	PUSHL	#COB\$ FAIGET_VM	
	00000000G	00		03 FB 0005E	CALLS	#3, LIB\$STOP	
		58	08	AE D0 00065	MOVL	DESC+4, ADDR	2216
		59		6C 9A 00069	MOVZBL	(AP), R9	2218
		57		01 D0 0006C	MOVL	#1, I	
				0F 11 0006F	BRB	6\$	
		56		6C47 D0 00071	MOVL	(AP)[I], PTR	2222
68	04	B6		66 28 00075	MOVC3	(PTR), @4(PTR), (ADDR)	2223
		50		66 3C 0007A	MOVZWL	(PTR), R0	2224
		58		50 C0 0007D	ADDL2	R0, ADDR	
	ED	57		59 F3 00080	AOBLEQ	R9, I, 5\$	2218
04	AE	58	08	AE A3 00084	SUBW3	DESC+4, ADDR, DESC	2227
			04	AE 9F 0008A	PUSHAB	DESC	2230
			04	AC DD 0008D	PUSHL	UNIT	
	0000V	CF		02 FB 00090	CALLS	#2, COMMON_CODE	
	00000084	8F		6E D1 00095	CMPL	COUNT, #132	2235
				0D 15 0009C	BLEQ	7\$	
			08	AE DD 0009E	PUSHL	DESC+4	2237
			04	AE 9F 000A1	PUSHAB	COUNT	
	00000000G	00		02 FB 000A4	CALLS	#2, LIB\$FREE_VM	
				04 000AB	RET		2239

; Routine Size: 172 bytes, Routine Base: _COB\$CODE + 0142


```

: 731      2240 1 ROUTINE COMMON_CODE (UNIT, STRING): NOVALUE =
: 732      2241 1
: 733      2242 1 ++
: 734      2243 1 FUNCTIONAL DESCRIPTION:
: 735      2244 1
: 736      2245 1     Performs processing which is common to both DISPLAY and
: 737      2246 1     DISPLAY WITH NO ADVANCING.
: 738      2247 1     consisting of:
: 739      2248 1         Open unit if currently not open
: 740      2249 1         Complete calculation of upspace code
: 741      2250 1         Writes out the string
: 742      2251 1
: 743      2252 1
: 744      2253 1
: 745      2254 1 FORMAL PARAMETERS:
: 746      2255 1
: 747      2256 1     UNIT.rl.v     integer unit number designating the device
: 748      2257 1                     on which the string(s) is(are) to be displayed.
: 749      2258 1
: 750      2259 1     STRING.rt.dx  address of descriptor for the concatenated
: 751      2260 1                     strings.
: 752      2261 1
: 753      2262 1 IMPLICIT INPUTS:
: 754      2263 1
: 755      2264 1     Status information as to whether the output file in question
: 756      2265 1     is currently open.
: 757      2266 1
: 758      2267 1 IMPLICIT OUTPUTS:
: 759      2268 1
: 760      2269 1     Updated status information for this file.
: 761      2270 1
: 762      2271 1 ROUTINE VALUE:
: 763      2272 1
: 764      2273 1     NONE
: 765      2274 1
: 766      2275 1 SIDE EFFECTS:
: 767      2276 1
: 768      2277 1     Outputs a record on the specified file.
: 769      2278 1
: 770      2279 1 --
: 771      2280 2 BEGIN
: 772      2281 2 MAP
: 773      2282 2     STRING: REF BLOCK[8, BYTE];
: 774      2283 2
: 775      2284 2 LOCAL
: 776      2285 2     FILE_NAME:     BLOCK [8,BYTE],      ! dynamically constructed desc.
: 777      2286 2     RAB:         REF $RAB_DECL;
: 778      2287 2
: 779      2288 2 LITERAL
: 780      2289 2     INIT_VALUE = 9 ;                      ! Initial COB$$AB_PREV value
: 781      2290 2
: 782      2291 2 IF .UNIT GTRU COB$K_UNIT_MAX
: 783      2292 2 THEN
: 784      2293 2     LIB$STOP(COB$_INVARG);
: 785      2294 2
: 786      2295 2
: 787      2296 2     ! If file is not yet open, open it.
```



```
788 2297 2 !
789 2298 2 !
790 2299 2 !
791 2300 2 !
792 2301 2 !
793 2302 2 !
794 2303 2 !
795 2304 2 !
796 2305 2 !
797 2306 2 !
798 2307 2 !
799 2308 2 !
800 2309 2 !
801 2310 2 !
802 2311 2 !
803 2312 2 !
804 2313 2 !
805 2314 2 !
806 2315 2 !
807 2316 2 !
808 2317 2 !
809 2318 2 !
810 2319 2 !
811 2320 2 !
812 2321 2 !
813 2322 2 !
814 2323 2 !
815 2324 2 !
816 2325 2 !
817 2326 2 !
818 2327 2 !
819 2328 2 !
820 2329 2 !
821 2330 2 !
822 2331 2 !
823 2332 2 !
824 2333 1 !

! IF .COB$$AL_WRITE_RAB[.UNIT] EQL 0
THEN
  +
  ! Second parameter of 0 signifies that COB$$OPEN_OUT is called on
  ! behave of VAX COBOL.
  -
  COB$$OPEN_OUT(.UNIT, 0);

! Calculate the upspacing codes needed to use on this action
! If previous operation was a DISPLAY, a line-feed is needed
COB$$AB_USPCODE[0] = 0;
IF .COB$$AB_PREV[0] EQL DISP OR .COB$$AB_PREV[0] EQL POS OR .COB$$AB_PREV[0] EQL ACC_ADV
OR .COB$$AB_PREV[0] EQL INIT_VALUE
THEN
  COB$$AB_USPCODE[0] = LINE_FEED;

! Write out the concatenated string
RAB = .COB$$AL_WRITE_RAB[.UNIT];
RAB[RAB$L_RBF] = .STRING[DSC$A_POINTER];
RAB[RAB$W_RSZ] = .STRING[DSC$W_LENGTH];

! Write the record. Retry certain errors, signal others.
WHILE $PUT(RAB = .RAB) EQL RMSS_RSA DO $WAIT(RAB = .RAB);

IF NOT .RAB[RAB$L_STS]
THEN
  LIB$STOP(COB$_ERRDURDIS, 1, .RAB+RAB$C_BLN, .RAB[RAB$L_STS], .RAB[RAB$L_STV]);
END;
```

.EXTRN SYSS\$PUT, SYSS\$WAIT

```
001C 00000 COMMON_CODE:
54 00000000G 00 9E 00002 .WORD Save R2,R3,R4
53 00000000' EF 9E 00009 MOVAB LIB$STOP, R4
5E 08 C2 00010 MOVAB COB$$AL_WRITE_RAB, R3
52 04 AC D0 00013 SUBL2 #8, SP
06 52 D1 00017 MOVL UNIT, R2
09 1B 0001A CMPL R2, #6
00000000G 8F DD 0001C BLEQU 1$
64 01 FB 00022 PUSHL #COB$ INVARG
6342 D5 00025 1$: CALLS #1, LIB$STOP
09 12 00028 TSTL COB$$AL_WRITE_RAB[R2]
7E D4 0002A BNEQ 2$
52 DD 0002C CLRL -(SP)
0000V CF 02 FB 0002E PUSHL R2
CALLS #2, COB$$OPEN_OUT
```

```
2240
2291
2293
2298
2304
```

		2C	A3	94	00033	2\$:	CLRB	COB\$\$AB_USPCODE	:	2310	
	50	30	A3	9A	00036		MOVZBL	COB\$\$AB_PREV, R0	:	2311	
			0F	13	0003A		BEQL	3\$	:		
	02		50	91	0003C		CMPB	R0, #2	:		
			0A	13	0003F		BEQL	3\$	:		
	04		50	91	00041		CMPB	R0, #4	:		
			05	13	00044		BEQL	3\$	:		
	09		50	91	00046		CMPB	R0, #9	:	2312	
			05	12	00049		BNEQ	4\$	:		
	2C	A3	8A	8F	90	0004B	3\$:	MOVB	#-118, COB\$\$AB_USPCODE	:	2314
		52	63	42	D0	00050	4\$:	MOVL	COB\$\$AL_WRITE_RAB[R2], RAB	:	2319
		50	08	AC	D0	00054		MOVL	STRING, R0	:	2320
	28	A2	04	A0	D0	00058		MOVL	4(R0), 40(RAB)	:	
	22	A2		60	B0	0005D		MOVW	(R0), 34(RAB)	:	2321
				52	DD	00061	5\$:	PUSHL	RAB	:	2326
00000000G	00			01	FB	00063		CALLS	#1, SYSSPUT	:	
000182DA	8F			50	D1	0006A		CMPL	R0, #99034	:	
				0B	12	00071		BNEQ	6\$	:	
				52	DD	00073		PUSHL	RAB	:	
00000000G	00			01	FB	00075		CALLS	#1, SYSSWAIT	:	
				E3	11	0007C		BRB	5\$	:	
	12	08	A2	E8	0007E	6\$:	BLBS	8(RAB), 7\$	:	2329	
	7E	08	A2	7D	00082		MOVQ	8(RAB), -(SP)	:	2331	
		44	A2	9F	00086		PUSHAB	68(RAB)	:		
				01	DD	00089		PUSHL	#1	:	
		00000000G		8F	DD	0008B		PUSHL	#COB\$ ERRDURDIS	:	
	64			05	FB	00091		CALLS	#5, LIB\$STOP	:	
				04	00094	7\$:	RET		:	2333	

; Routine Size: 149 bytes, Routine Base: _COB\$CODE + 01EE


```
826 2334 1 GLOBAL ROUTINE COB$$OPEN_OUT (UNIT, RPG): NOVALUE =
827 2335 1
828 2336 1 ++
829 2337 1 FUNCTIONAL DESCRIPTION:
830 2338 1
831 2339 1     Open a file for writing, given a unit number.
832 2340 1
833 2341 1 FORMAL PARAMETERS:
834 2342 1
835 2343 1     UNIT.rl.v     integer unit number designating the device
836 2344 1                on which the string(s) is(are) to be displayed.
837 2345 1
838 2346 1     RPG.rl.v      = 1 if COB$$OPEN_OUT called for VAX RPG
839 2347 1                = 0 if COB$$OPEN_OUT called for VAX COBOL
840 2348 1
841 2349 1 IMPLICIT INPUTS:
842 2350 1
843 2351 1     NONE
844 2352 1
845 2353 1 IMPLICIT OUTPUTS:
846 2354 1
847 2355 1     NONE
848 2356 1
849 2357 1 ROUTINE VALUE:
850 2358 1
851 2359 1     NONE
852 2360 1
853 2361 1 SIDE EFFECTS:
854 2362 1
855 2363 1     Opens a file.  On error, Signals a fatal condition.
856 2364 1
857 2365 1 --
858 2366 1
859 2367 2 BEGIN
860 2368 2 LITERAL
861 2369 2     MAX_BUF =      MAX(64, NAM$C_MAXRSS);
862 2370 2 LOCAL
863 2371 2     FAB:          $FAB_DECL,
864 2372 2     NAM:          $NAM_DECL,
865 2373 2     RAB:          REF $RAB_DECL,
866 2374 2     FILE_NAME:    BLOCK[8, BYTE],      ! Descriptor for the file name
867 2375 2     TRANSLATE:     BLOCK[8, BYTE],
868 2376 2     P:             REF VECTOR[BYTE],
869 2377 2     RSLBUF:        VECTOR[MAX_BUF, BYTE],
870 2378 2     STATUS;
871 2379 2
872 2380 2
873 2381 2 ! Determine whether the COB$xxx name is defined.
874 2382 2 ! If so, use it.  If not, use the corresponding SYSS$xxx name.
875 2383 2
876 2384 2 TRANSLATE[DSC$B_DTYPE] = DSC$K_DTYPE_T;
877 2385 2 TRANSLATE[DSC$B_CLASS] = DSC$K_CLASS_S;
878 2386 2 TRANSLATE[DSC$W_LENGTH] = MAX_BUF;
879 2387 2 TRANSLATE[DSC$A_POINTER] = RSLBUF;
880 2388 2
881 2389 2 ++
882 2390 2 ! If VAX RPG is calling this routine, bypass COB_TABLE.
```

```
883 2391 2 !-
884 2392 2
885 2393 2 IF .RPG EQL 1
886 2394 2 THEN
887 2395 2 BEGIN ! Use the SYS$xxx logical
888 2396 2 P = .SYS TABLE[.UNIT] + BASE;
889 2397 2 FILE_NAME[DSC$W_LENGTH] = .P[0];
890 2398 2 FILE_NAME[DSC$A_POINTER] = P[1];
891 2399 2 END
892 2400 2 ELSE
893 2401 2 BEGIN ! Use the COB$xxx logical
894 2402 2 P = .COB TABLE[.UNIT] + BASE;
895 2403 2 FILE_NAME[DSC$B_DTYPE] = DSC$K_DTYPE_T;
896 2404 2 FILE_NAME[DSC$B_CLASS] = DSC$K_CLASS_S;
897 2405 2 FILE_NAME[DSC$W_LENGTH] = .P[0];
898 2406 2 FILE_NAME[DSC$A_POINTER] = P[1];
899 2407 2 IF $TRNLOG(LOGNAM = FILE_NAME, RSLBUF = TRANSLATE) NEQ SS$_NORMAL
900 2408 2 THEN
901 2409 2 BEGIN ! Use the SYS$xxx logical
902 2410 2 P = .SYS TABLE[.UNIT] + BASE;
903 2411 2 FILE_NAME[DSC$W_LENGTH] = .P[0];
904 2412 2 FILE_NAME[DSC$A_POINTER] = P[1];
905 2413 2 END;
906 2414 2 END ;
907 2415 2
908 2416 2
909 P 2417 2 $FAB INIT(
910 P P 2418 2 FAB = FAB,
911 P P 2419 2 NAM = NAM,
912 P P 2420 2 FAC = PUT,
913 P P 2421 2 FNA = .FILE_NAME[DSC$A_POINTER],
914 P P 2422 2 FNS = .FILE_NAME[DSC$W_LENGTH],
915 P P 2423 2 RAT = PRN,
916 P 2424 2 FOP = SQO,
917 2425 2 RFM = VFC);
918 2426 2
919 P 2427 2 $NAM INIT(
920 P P 2428 2 NAM = NAM,
921 P P 2429 2 ESA = RSLBUF,
922 P P 2430 2 ESS = NAM$C_MAXRSS,
923 P 2431 2 RSA = RSLBUF,
924 2432 2 RSS = NAM$C_MAXRSS);
925 2433 2
926 2434 2 STATUS = $CREATE(FAB = FAB);
927 2435 2 IF (TRANSLATE[DSC$W_LENGTH] = .NAM[NAM$B_RSL]) EQL 0 THEN
928 2436 2 IF (TRANSLATE[DSC$W_LENGTH] = .NAM[NAM$B_ESL]) EQL 0
929 2437 2 THEN
930 2438 2 BEGIN
931 2439 2 TRANSLATE[DSC$W_LENGTH] = .FAB[FAB$B_FNS];
932 2440 2 TRANSLATE[DSC$A_POINTER] = .FAB[FAB$L_FNA];
933 2441 2 END;
934 2442 2
935 2443 2
936 2444 2 IF NOT .STATUS
937 2445 2 THEN
938 2446 2 LIB$STOP(COB$_ERRDURDIS, 1, TRANSLATE, .FAB[FAB$L_STS], .FAB[FAB$L_STV]);
939 2447 2
```



```

940 2448 2
941 2449
942 2450
943 2451
944 2452
945 2453
946 2454
947 2455
948 2456
949 2457
950 2458
951 2459
952 2460
953 2461
954 2462
955 2463
956 2464
957 2465
958 2466
959 2467
960 2468
961 2469
962 2470
963 2471
964 2472
965 2473
966 2474
967 2475
968 2476
969 2477
970 2478
971 2479
972 2480
973 2481 1

IF NOT (STATUS = LIB$GET_VM(%REF(RAB$C_BLN + 8 + .NAM[NAM$B_RSL]), RAB))
THEN
    LIB$STOP(COB$_FAIGET_VM, 0, .STATUS);

! Save a descriptor for the resultant file name string,
! and the string itself, after the RAB
BEGIN
LOCAL
    Q: REF BLOCK[.BYTE];
    Q = .RAB + RAB$C_BLN;
    Q[DSC$B_DTYPE] = DSC$K_DTYPE_T;
    Q[DSC$B_CLASS] = DSC$K_CLASS_S;
    Q[DSC$W_LENGTH] = .TRANSLATE[DSC$W_LENGTH];
    Q[DSC$A_POINTER] = .RAB+RAB$C_BLN+8;
    CH$MOVE7 .Q[DSC$W_LENGTH], .TRANSLATE[DSC$A_POINTER], .RAB+RAB$C_BLN+8 );
END;

$RAB_INIT(
    RAB = .RAB,
    FAB = FAB,
    ROP = EOF,
    RHB = COB$$AB_USPCODE);

IF NOT $CONNECT(RAB = .RAB)
THEN
    LIB$STOP(COB$_ERRDURDIS, 1, .RAB+RAB$C_BLN, .RAB[RAB$L_STS], .RAB[RAB$L_STV]);

COB$$AL_WRITE_RAB[.UNIT] = .RAB;
COB$$AW_WRITE_IFI[.UNIT] = .FAB[FAB$W_IFI];
END;

! end of COB$$OPEN_OUT
```

				OFFC 00000	.EXTRN	SYS\$TRNLOG, SYS\$CREATE	
					.EXTRN	SYS\$CONNECT	
					.ENTRY	COB\$\$OPEN_OUT, Save R2,R3,R4,R5,R6,R7,R8,-	2334
						R9,R10,R11	
					MOVAB	COB\$\$AB_USPCODE, R11	
					MOVAB	LIB\$STOP, R10	
					MOVAB	-456(SP), SP	
					MOVL	#17694975, TRANSLATE	2386
					MOVAB	RSLBUF, TRANSLATE+4	2387
					MOVL	UNIT, R8	2396
					ASHL	#2, R8, R7	
					CMPL	RP6, #1	2393
					BEQL	1\$	
					MOVAB	BASE, R0	2402
					PUSHAB	COB_TABLE[R7]	
					ADDL3	a(SP)+, R0, P	
					MOVW	#270, FILE_NAME+2	2403
					MOVZBW	(P), FILE_NAME	2405
					MOVAB	1(R2), FICE_NAME+4	2406

57	FF40	CD	010E00FF	8F	D0	00015	
	FF44	CD	08	AE	9E	0001E	
		58	04	AC	D0	00024	
		58		02	78	00028	
		01	08	AC	D1	0002C	
				3A	13	00030	
		50	FD47	CF	9E	00032	
			FDA9	CF47	9F	00037	
52		50		9E	C1	0003C	
	FF4A	CD	010E	8F	B0	00040	
	FF48	CD		62	9B	00047	
	FF4C	CD	01	A2	9E	0004C	

Address	Hex	Assembly	Comment	Address	Hex	Assembly	Comment
0050	8F	00	00000000G	7E	7C	00052	CLRQ -(SP)
				7E	D4	00054	CLRL -(SP)
				CD	9F	00056	PUSHAB TRANSLATE
				7E	D4	0005A	CLRL -(SP)
				CD	9F	0005C	PUSHAB FILE_NAME
				06	FB	00060	CALLS #6, SYS\$STRNLOG
				50	D1	00067	CMPL R0, #1
				19	13	0006A	BEQL 2\$
				CF	9E	0006C	MOVAB BASE, R0
				CF	9F	00071	PUSHAB SYS_TABLE[R7]
				9E	C1	00076	ADDL3 @ (SP)+, R0, P
				62	9B	0007A	MOVZBW (P), FILE_NAME
				A2	9E	0007F	MOVAB 1(R2), FILE_NAME+4
				00	2C	00085	MOVCS #0, (SP), #0, #80, \$RMS_PTR
				AD		0008C	
				8F	B0	0008E	MOVW #20483, \$RMS_PTR
				8F	9A	00094	MOVZBL #64, \$RMS_PTR+4
				01	90	00099	MOVB #1, \$RMS_PTR+22
				8F	B0	0009D	MOVW #772, \$RMS_PTR+30
				CD	9E	000A3	MOVAB NAM, \$RMS_PTR+40
				CD	D0	000A9	MOVL FILE_NAME+4, \$RMS_PTR+44
				CD	90	000AF	MOVB FILE_NAME, \$RMS_PTR+52
				00	2C	000B5	MOVCS #0, (SP), #0, #96, \$RMS_PTR
				CD		000BC	
				8F	B0	000BF	MOVW #24578, \$RMS_PTR
				01	8E	000C6	MNEGB #1, \$RMS_PTR+2
				AE	9E	000CB	MOVAB RSLBUF, \$RMS_PTR+4
				01	8E	000D1	MNEGB #1, \$RMS_PTR+10
				AE	9E	000D6	MOVAB RSLBUF, \$RMS_PTR+12
				AD	9F	000DC	PUSHAB FAB
				01	FB	000DF	CALLS #1, SYS\$CREATE
				50	D0	000E6	MOVL R0, STATUS
				CD	9B	000E9	MOVZBW NAM+3, TRANSLATE
				15	12	000F0	BNEQ 3\$
				CD	9B	000F2	MOVZBW NAM+11, TRANSLATE
				0C	12	000F9	BNEQ 3\$
				AD	9B	000FB	MOVZBW FAB+52, TRANSLATE
				AD	D0	00101	MOVL FAB+44, TRANSLATE+4
				52	E8	00107	BLBS STATUS, 4\$
				AD	7D	0010A	MOVQ FAB+8, -(SP)
				CD	9F	0010E	PUSHAB TRANSLATE
				01	DD	00112	PUSHL #1
				8F	DD	00114	PUSHL #COB\$ERRDURDIS
				05	FB	0011A	CALLS #5, LIB\$STOP
				AE	9F	0011D	PUSHAB RAB
				CD	9A	00120	MOVZBL NAM+3, 4(SP)
				8F	C0	00126	ADDL2 #76, 4(SP)
				AE	9F	0012E	PUSHAB 4(SP)
				02	FB	00131	CALLS #2, LIB\$GET_VM
				50	D0	00138	MOVL R0, STATUS
				52	E8	0013B	BLBS STATUS, 5\$
				52	DD	0013E	PUSHL STATUS
				7E	D4	00140	CLRL -(SP)
				8F	DD	00142	PUSHL #COB\$FAIGET_VM
				03	FB	00148	CALLS #3, LIB\$STOP
				AE	D0	0014B	MOVL RAB, R6
				A6	9E	0014F	MOVAB 68(R6), R9

0044	8F	4C	A6	00	FF44	50	02	A0	010E	59	D0	00153	MOVL	R9, 0	2461	
						60	04	A0	FF40	8F	B0	00156	MOVW	#270, 2(Q)	2463	
						DD			4C	CD	B0	0015C	MOVW	TRANSLATE, (Q)	2464	
						6E				A6	9E	00161	MOVAB	76(R6), 4(Q)	2465	
										60	28	00166	MOVCS	(Q), @TRANSLATE+4, 76(R6)	2473	
										00	2C	0016D	MOVCS	#0, (SP), #0, #68, (R6)		
										66		00174				
										8F	B0	00175	MOVW	#17409, (R6)		
										8F	3C	0017A	MOVZWL	#256, 4(R6)		
										6B	9E	00180	MOVAB	COB\$\$AB_USPCODE, 44(R6)		
										AD	9E	00184	MOVAB	FAB, 60(R6)		
										56	DD	00189	PUSHL	R6	2475	
										01	FB	0018B	CALLS	#1, SYSS\$CONNECT		
										50	E8	00192	BLBS	R0, 6\$		
										A6	7D	00195	MOVQ	8(R6), -(SP)	2477	
										59	DD	00199	PUSHL	R9		
										01	DD	0019B	PUSHL	#1		
										8F	DD	0019D	PUSHL	#COB\$ ERRDURDIS		
										05	FB	001A3	CALLS	#5, LIB\$STOP		
										D4	AB47	9F	001A6	PUSHAB	COB\$\$AL WRITE_RAB[R7]	2479
										56	D0	001AA	MOVL	R6, @ (SP)+		
										B0	001AD	MOVW	FAB+2, COB\$\$AW_WRITE_IFI[R8]	2480		
										04	001B3	RET		2481		

; Routine Size: 436 bytes, Routine Base: _COB\$CODE + 0283

```

975 2482 1 ROUTINE COMMON_SCREEN (
976 2483 1     UNIT,
977 2484 1     STRING : REF BLOCK [8,BYTE],
978 2485 1     FLAGS
979 2486 1     ): NOVALUE =
980 2487 1
981 2488 1
982 2489 1 ++
983 2490 1 FUNCTIONAL DESCRIPTION:
984 2491 1     Performs processing which is common to both COB$DISP_SCR and
985 2492 1     COB$DISP_SCR_NO_ADV.
986 2493 1     This includes :
987 2494 1         Open unit if currently not open
988 2495 1         Complete calculation of upspace code
989 2496 1         Call conversion routine DISP_CONVERT
990 2497 1         Call COB$$$SETUP_TERM_TYPE
991 2498 1         Call COB$$$SET_ATTRIBUTES
992 2499 1         Write out the string
993 2500 1
994 2501 1
995 2502 1 FORMAL PARAMETERS:
996 2503 1
997 2504 1     UNIT.rl.v     integer unit number designating the device
998 2505 1                 on which the string is to be displayed.
999 2506 1
1000 2507 1     STRING.rt.dx   address of string descriptor which is to be
1001 2508 1                 displayed on the specified device.
1002 2509 1
1003 2510 1     FLAGS.rlu.v    screen enhancement flag;
1004 2511 1
1005 2512 1                 bit 0 - bold
1006 2513 1                 bit 1 - reverse
1007 2514 1                 bit 2 - blinking
1008 2515 1                 bit 3 - underline
1009 2516 1                 bit 4 - bell
1010 2517 1                 bit 5 - conversion
1011 2518 1                 bit 6 - decimal point is comma
1012 2519 1                 bit 7 - 0 print sign, 1 do not print sign
1013 2520 1                 bit 11 - 0 for VAX COBOL, 1 for VAX RPG
1014 2521 1
1015 2522 1 IMPLICIT INPUTS:
1016 2523 1
1017 2524 1     Status information as to whether the output file in question
1018 2525 1     is currently open.
1019 2526 1
1020 2527 1 IMPLICIT OUTPUTS:
1021 2528 1
1022 2529 1     Updated status information for this file.
1023 2530 1
1024 2531 1 ROUTINE VALUE:
1025 2532 1
1026 2533 1     NONE
1027 2534 1
1028 2535 1 SIDE EFFECTS:
1029 2536 1
1030 2537 1     Outputs a record on the specified file.
1031 2538 1
```



```
1032 2539 1 !-- BEGIN
1033 2540 2
1034 2541 2
1035 2542 2 BUILTIN
1036 2543 2 ACTUALPARAMETER,
1037 2544 2 ACTUALCOUNT;
1038 2545 2
1039 2546 2 LOCAL
1040 2547 2   ANS_STRING : BLOCK [8,BYTE],      ! Descriptor for output
1041 2548 2   PUT_FLAG   : INITIAL (0),        ! Longword flag for $PUT
1042 2549 2   FAB       : REF $FAB_DECL,      ! Fab for output device
1043 2550 2   RAB       : REF $RAB_DECL,     ! Rab for output device
1044 2551 2   OUT_BUF    : VECTOR [COB$K_ACC_SIZE,BYTE], ! Buffer passed to
1045 2552 2   !                                     COB$$$SET_ATTRIBUTES
1046 2553 2   OUT_LEN     : INITIAL (0);        ! Passed to COB$$$SET_ATTRIBUTES
1047 2554 2
1048 2555 2 LITERAL
1049 2556 2   INIT_VALUE = 9;                ! Initial COB$$AB_PREV value
1050 2557 2
1051 2558 2 !+
1052 2559 2 ! There should be no more than 3 parameters
1053 2560 2 !-
1054 2561 2
1055 2562 2 IF ACTUALCOUNT() GTR 3
1056 2563 2 THEN
1057 2564 2   LIB$STOP(COB$_INVARG);
1058 2565 2
1059 2566 2 IF .UNIT GTRU COB$K_UNIT_MAX
1060 2567 2 THEN
1061 2568 2   LIB$STOP(COB$_INVARG);
1062 2569 2
1063 2570 2 !+
1064 2571 2 ! If file is not yet open, open it.
1065 2572 2 !-
1066 2573 2
1067 2574 2 IF .COB$$AL_WRITE_RAB[.UNIT] EQL 0
1068 2575 2 THEN
1069 2576 2   !+
1070 2577 2   ! Second parameter tells COB$$OPEN_OUT whether VAX COBOL (0)
1071 2578 2   ! or VAX RPG (1) is the caller.
1072 2579 2   !-
1073 2580 2   COB$$OPEN_OUT ( .UNIT,
1074 2581 2   !                                     IF ( .FLAGS AND V_COB_RPG ) NEQ 0
1075 2582 2   !                                     THEN 1
1076 2583 2   !                                     ELSE 0 );
1077 2584 2
1078 2585 2 !+
1079 2586 2 ! Calculate the upspacing codes we need to use on this action.
1080 2587 2 ! If previous operation was a DISPLAY (COB$DISPLAY or COB$DISP_SCR),
1081 2588 2 ! a line-feed is needed.
1082 2589 2 !-
1083 2590 2 COB$$AB_USPCODE[0] = 0;
1084 2591 2 IF .COB$$AB_PREV[0] EQL DISP OR .COB$$AB_PREV[0] EQL POS OR .COB$$AB_PREV[0] EQL ACC_ADV
1085 2592 2   OR .COB$$AB_PREV[0] EQL INIT_VALUE
1086 2593 2 THEN
1087 2594 2   COB$$AB_USPCODE[0] = LINE_FEED;
1088 2595 2
```

```
1089 2596 2
1090 2597
1091 2598
1092 2599
1093 2600
1094 2601
1095 2602
1096 2603
1097 2604
1098 2605
1099 2606
1100 2607
1101 2608
1102 2609
1103 2610
1104 2611
1105 2612
1106 2613
1107 2614
1108 2615
1109 2616
1110 2617
1111 2618
1112 2619
1113 2620
1114 2621
1115 2622
1116 2623
1117 2624
1118 2625
1119 2626
1120 2627
1121 2628
1122 2629
1123 2630
1124 2631
1125 2632
1126 2633
1127 2634
1128 2635
1129 2636
1130 2637
1131 2638
1132 2639
1133 2640
1134 2641
1135 2642
1136 2643
1137 2644
1138 2645
1139 2646
1140 2647
1141 2648
1142 2649
1143 2650
1144 2651
1145 2652 2

+ Create descriptor ANS_STRING. All TYPEs and CLASSEs of input
string descriptors will eventually be deposited (through conversion
and parsing) into ANS_STRING for output.
Because STR$COPY_R is used there is no need to allocate and
deallocate space for ANS_STRING as STR$COPY_R will do this.
-

ANS_STRING [DSC$W_LENGTH] = 0 ;
ANS_STRING [DSC$B_DTYPE] = DSC$K_DTYPE_T ;
ANS_STRING [DSC$B_CLASS] = DSC$K_CLASS_D ;
ANS_STRING [DSC$A_POINTER] = 0 ;

+ Check FLAGS parameter. If conversion requested (bit 5),
call routine to convert and parse the various data types.
Convert all data types to Text.
-

IF ( .FLAGS AND V_CONV ) NEQ 0
THEN DISP_CONVERT ( .STRING, .FLAGS, ANS_STRING )

+ This will handle TEXT without CONVERSION and anything else
without conversion.
Note - if user does not request conversion for any data type,
the string will be output as is (same results as old DISPLAY).
-

ELSE
IF NOT (STR$COPY_R ( ANS_STRING, STRING[DSC$W_LENGTH],
.STRING [DSC$A_POINTER] ))
THEN LIB$STOP (COB$ERRDURDIS) ;

+ Conversion and Parsing completed (if requested) - Display string.
Break down FLAGS parameter to a valid parameter for $PUT.
(ie. the first four bits (0-3) of FLAGS parameter are passed to $PUT)
Determine whether or not to ring terminal bell (bit 4).
-

PUT_FLAG = .FLAGS AND FLAG_MASK ;
IF ( .FLAGS AND V_BELL ) NEQ 0
THEN
BEGIN
OUT_BUF[0] = BELL ;
OUT_LEN = .OUT_LEN + 1 ;
END ;

+ Request for bold, reverse, blinking, underline, or any combination
thereof. It is first necessary to determine terminal type.
COB$$SETUP_TERM_TYPE puts this information in COB$TERM_TYPE.
Call COB$$SET_ATTRIBUTES to turn on requested terminal attributes.
After call OUT_BUF contains concatenation of -
bell sequence, if requested
escape sequence to turn on attributes,
```



```

: 1146      2653      2      |      final form of input string to be displayed, and
: 1147      2654      2      |      escape sequence to turn off attributes.
: 1148      2655      2      |      OUT_LEN is updated in COB$$$SET_ATTRIBUTES.
: 1149      2656      2      |      COB$$$SET_ATTRIBUTES is called even if no terminal attributes
: 1150      2657      2      |      are requested to copy ANS_STRING to OUT_BUF.
: 1151      2658      2      |      -
: 1152      2659      2      |
: 1153      2660      2      |      RAB = .COB$$AL WRITE RAB [.UNIT];
: 1154      2661      2      |      IF .COB$TERM_TYPE EQ 0      |      ! If terminal type not
: 1155      2662      2      |      THEN      |      ! yet determined
: 1156      2663      2      |      BEGIN
: 1157      2664      2      |      LOCAL
: 1158      2665      2      |      NAM_DSC : REF BLOCK [,BYTE] ;      |      ! Name dsc (from
: 1159      2666      2      |      |      COB$$OPEN_OUT)
: 1160      2667      2      |      NAM_DSC = .RAB + RAB$C_BLN ;
: 1161      2668      2      |      IF NOT ( COB$$$SETUP_TERM_TYPE ( .NAM_DSC [DSC$A_POINTER],
: 1162      2669      2      |      |      .NAM_DSC [DSC$W_LENGTH],
: 1163      2670      2      |      |      COB$TERM_TYPE ) )
: 1164      2671      2      |      THEN LIB$STOP (COB$_ERRDURDIS) ;
: 1165      2672      2      |
: 1166      2673      2      |      IF .COB$TERM_TYPE EQL UNKNOWN
: 1167      2674      2      |      THEN
: 1168      2675      2      |      COB$TERM_TYPE = VT100;      |      ! treat file as VT100
: 1169      2676      2      |
: 1170      2677      2      |      END ;
: 1171      2678      2      |
: 1172      2679      2      |      IF NOT ( COB$$$SET_ATTRIBUTES ( .COB$TERM_TYPE, .ANS_STRING [DSC$A_POINTER],
: 1173      2680      2      |      |      .ANS_STRING [DSC$W_LENGTH], .PUT_FLAG,
: 1174      2681      2      |      |      OUT_BUF[0], OUT_LEN ) )
: 1175      2682      2      |      THEN LIB$STOP (COB$_ERRDURDIS) ;
: 1176      2683      2      |
: 1177      2684      2      |      !+
: 1178      2685      2      |      ! Put OUT_BUF in RAB
: 1179      2686      2      |      !-
: 1180      2687      2      |
: 1181      2688      2      |      RAB [RAB$L_RBF] = OUT_BUF [0] ;
: 1182      2689      2      |      RAB [RAB$W_RSZ] = .OUT_LEN ;
: 1183      2690      2      |
: 1184      2691      2      |      !+
: 1185      2692      2      |      ! Display the final form of the original input string.
: 1186      2693      2      |      !-
: 1187      2694      2      |
: 1188      2695      2      |      WHILE $PUT(RAB = .RAB) EQL RMSS_RSA DO $WAIT(RAB = .RAB) ;
: 1189      2696      2      |
: 1190      2697      2      |      IF NOT .RAB [RAB$L_STS]
: 1191      2698      2      |      THEN
: 1192      2699      2      |      LIB$STOP (COB$_ERRDURDIS, 1, .RAB+RAB$C_BLN, .RAB[RAB$L_STS],
: 1193      2700      2      |      |      .RAB[RAB$L_STV]) ;
: 1194      2701      2      |
: 1195      2702      1      |      END ;      |      ! end COMMON_SCREEN
```

58	00000000G	8F	DO	00002	MOVL	#COB\$-INVARG, R8			
57	00000000G	8F	DO	00009	MOVL	#COB\$-ERRDURDIS, R7			
56	00000000G	00	9E	00010	MOVAB	COB\$TERM_TYPE, R6			
55	00000000G	EF	9E	00017	MOVAB	COB\$\$AL_WRITE_RAB, R5			
54	00000000G	00	9E	0001E	MOVAB	LIB\$STOP, R4			
5E	FBF8	CE	9E	00025	MOVAB	-1032(SP), SP			
		53	D4	0002A	CLRL	PUT_FLAG	2540		
		7E	D4	0002C	CLRL	OUT_LEN			
03		6C	91	0002E	CMPB	(APT), #3	2562		
		05	1B	00031	BLEQU	1\$			
		58	DD	00033	PUSHL	R8	2564		
64		01	FB	00035	CALLS	#1, LIB\$STOP			
52	04	AC	DO	00038	1\$: MOVL	UNIT, R2	2566		
06		52	D1	0003C	CML	R2, #6			
		05	1B	0003F	BLEQU	2\$			
		58	DD	00041	PUSHL	R8	2568		
64		01	FB	00043	CALLS	#1, LIB\$STOP			
		6542	D5	00046	2\$: TSTL	COB\$\$AL_WRITE_RAB[R2]	2574		
		12	12	00049	BNEQ	5\$			
04	OC	AC	0B	E1	0004B	BBC	#11, FLAGS, 3\$	2581	
		01	DD	00050	PUSHL	#1			
		02	11	00052	BRB	4\$			
		7E	D4	00054	3\$: CLRL	-(SP)			
		52	DD	00056	4\$: PUSHL	R2	2580		
FDEF	CF		02	FB	00058	CALLS	#2, COB\$\$OPEN OUT		
	2C		A5	94	0005D	5\$: CLRB	COB\$\$AB_USPCODE	2590	
50	30		A5	9A	00060	MOVZBL	COB\$\$AB_PREV, R0	2591	
			0F	13	00064	BEQL	6\$		
02			50	91	00066	CMPB	R0, #2		
			0A	13	00069	BEQL	6\$		
04			50	91	0006B	CMPB	R0, #4		
			05	13	0006E	BEQL	6\$		
09			50	91	00070	CMPB	R0, #9	2592	
	2C	A5	05	12	00073	BNEQ	7\$		
	F8	AD	8A	8F	90	00075	6\$: MOVAB	#-118, COB\$\$AB_USPCODE	2594
		020E0000	8F	DO	0007A	7\$: MOVL	#34471936, ANS_STRING	2604	
			FC	AD	D4	00082	CLRL	ANS_STRING+4	2607
			08	AC	DO	00085	MOVL	STRING, R0	2616
OF	OC	AC	05	E1	00089	BBC	#5, FLAGS, 8\$	2615	
			F8	AD	9F	0008E	PUSHAB	ANS_STRING	2616
			OC	AC	DD	00091	PUSHL	FLAGS	
			50	DD	00094	PUSHL	R0		
0000V	CF		03	FB	00096	CALLS	#3, DISP_CONVERT		
			17	11	0009B	BRB	9\$		
		04	A0	DD	0009D	8\$: PUSHL	4(R0)	2627	
			50	DD	000A0	PUSHL	R0	2626	
		F8	AD	9F	000A2	PUSHAB	ANS_STRING		
00000000G	00		03	FB	000A5	CALLS	#3, STR\$COPY_R		
05			50	E8	000AC	BLBS	R0, 9\$		
			57	DD	000AF	PUSHL	R7	2628	
			01	FB	000B1	CALLS	#1, LIB\$STOP		
53	OC	AC	00	EF	000B4	9\$: EXTZV	#0, #4, FLAGS, PUT_FLAG	2637	
		06	04	E1	000BA	BBC	#4, FLAGS, 10\$	2638	
			07	90	000BF	MOVAB	#7, OUT_BUF	2641	
			6E	D6	000C3	INCL	OUT_LEN	2642	
			52	6542	DO	000C5	10\$: MOVL	COB\$\$AL_WRITE_RAB[R2], RAB	2660
			66	D5	000C9	TSTL	COB\$TERM_TYPE	2661	

	50	44	22	12	000CB	BNEQ	12\$	:	2667
			A2	9E	000CD	MOVAB	68(R2), NAM_DSC	:	2668
	7E		56	DD	000D1	PUSHL	R6	:	2669
			60	3C	000D3	MOVZWL	(NAM_DSC), -(SP)	:	2668
00000000G	00	04	A0	DD	000D6	PUSHL	4(NAM_DSC)	:	
	05		03	FB	000D9	CALLS	#3, COB\$\$\$SETUP_TERM_TYPE	:	
			50	E8	000E0	BLBS	R0, 11\$	:	
	64		57	DD	000E3	PUSHL	R7	:	2671
			01	FB	000E5	CALLS	#1, LIB\$STOP	:	
			66	D5	000E8	TSTL	COB\$TERM_TYPE	:	2673
	66		03	12	000EA	BNEQ	12\$	:	
			03	D0	000EC	MOVL	#3, COB\$TERM_TYPE	:	2675
			5E	DD	000EF	PUSHL	SP	:	2681
		08	AE	9F	000F1	PUSHAB	OUT_BUF	:	
			53	DD	000F4	PUSHL	PUT_FLAG	:	2680
	7E	F8	AD	3C	000F6	MOVZWL	ANS_STRING, -(SP)	:	
		FC	AD	DD	000FA	PUSHL	ANS_STRING+4	:	2679
			66	DD	000FD	PUSHL	COB\$TERM_TYPE	:	
00000000G	00		06	FB	000FF	CALLS	#6, COB\$\$\$SET_ATTRIBUTES	:	
	05		50	E8	00106	BLBS	R0, 13\$	:	
			57	DD	00109	PUSHL	R7	:	2682
	64		01	FB	0010B	CALLS	#1, LIB\$STOP	:	
28	A2	04	AE	9E	0010E	MOVAB	OUT_BUF, 40(RAB)	:	2688
22	A2		6E	B0	00113	MOVW	OUT_LEN, 34(RAB)	:	2689
			52	DD	00117	PUSHL	RAB	:	2695
00000000G	00		01	FB	00119	CALLS	#1, SYSS\$PUT	:	
000182DA	8F		50	D1	00120	CMPL	R0, #99034	:	
			0B	12	00127	BNEQ	15\$	:	
			52	DD	00129	PUSHL	RAB	:	
00000000G	00		01	FB	0012B	CALLS	#1, SYSS\$WAIT	:	
			E3	11	00132	BRB	14\$	:	
	0E	08	A2	E8	00134	BLBS	8(RAB), 16\$	:	2697
	7E	08	A2	7D	00138	MOVQ	8(RAB), -(SP)	:	2699
		44	A2	9F	0013C	PUSHAB	68(RAB)	:	
			01	DD	0013F	PUSHL	#1	:	
			57	DD	00141	PUSHL	R7	:	
	64		05	FB	00143	CALLS	#5, LIB\$STOP	:	
			04	00146	16\$:	RET		:	2702

; Routine Size: 327 bytes, Routine Base: _COB\$CODE + 0437


```
: 1197      2703 1 ROUTINE DISP CONVERT (           ! Convert all data types to Text
: 1198      2704 1     STRING      : REF $STR$DESCRIPTOR, ! Input string
: 1199      2705 1     FLAGS      : REF $STR$DESCRIPTOR ! Screen enhancement flag
: 1200      2706 1     ANS_STRING : REF $STR$DESCRIPTOR ! Final form of string for output
: 1201      2707 1     ) : NOVALUE =
: 1202      2708 1
: 1203      2709 1 ++
: 1204      2710 1 FUNCTIONAL DESCRIPTION:
: 1205      2711 1
: 1206      2712 1     Convert the various VAX-11 COBOL data types to Text for output.
: 1207      2713 1     Call DISP_PARSE to add the final touches.
: 1208      2714 1
: 1209      2715 1
: 1210      2716 1 FORMAL PARAMETERS:
: 1211      2717 1
: 1212      2718 1     STRING.rt.dx    address of input string descriptor
: 1213      2719 1
: 1214      2720 1     FLAGS.rlu.v    screen enhancement flag (not used in this routine
: 1215      2721 1                               but passed to DISP_PARSE)
: 1216      2722 1                               bit 0 - bold
: 1217      2723 1                               bit 1 - reverse
: 1218      2724 1                               bit 2 - blinking
: 1219      2725 1                               bit 3 - underline
: 1220      2726 1                               bit 4 - bell
: 1221      2727 1                               bit 5 - conversion
: 1222      2728 1                               bit 6 - decimal point is comma
: 1223      2729 1                               bit 7 - 0 print sign, 1 do not print sign
: 1224      2730 1                               bit 11 - 0 for VAX COBOL, 1 for VAX RPG
: 1225      2731 1
: 1226      2732 1     ANS_STRING.wt.dx address of descriptor to hold final form of
: 1227      2733 1                               string to be displayed on specified device
: 1228      2734 1
: 1229      2735 1
: 1230      2736 1 IMPLICIT INPUTS:
: 1231      2737 1
: 1232      2738 1     NONE
: 1233      2739 1
: 1234      2740 1 IMPLICIT OUTPUTS:
: 1235      2741 1
: 1236      2742 1     Updated status information.
: 1237      2743 1
: 1238      2744 1 ROUTINE VALUE:
: 1239      2745 1
: 1240      2746 1     NONE
: 1241      2747 1
: 1242      2748 1 SIDE EFFECTS:
: 1243      2749 1
: 1244      2750 1     Converts all VAX-11 COBOL data types to TEXT.
: 1245      2751 1 --
: 1246      2752 1
: 1247      2753 2 BEGIN
: 1248      2754 2
: 1249      2755 2 EXTERNAL
: 1250      2756 2     LIB$AB_CVTTP_O,      ! for CVTTP
: 1251      2757 2     LIB$AB_CVTTP_U,      ! for CH$TRANSLATE
: 1252      2758 2     LIB$AB_CVT_O_U ;      ! for CVTTP after CH$TRANSLATE
: 1253      2759 2
```



```
: 1254      2760  2  BUILTIN
: 1255      2761  2  CVTTP,
: 1256      2762  2  CVTPS,
: 1257      2763  2  CVTLP ;
: 1258      2764  2
: 1259      2765  2  LOCAL
: 1260      2766  2  TEMP_DESC : BLOCK [12,BYTE] INITIAL (0) VOLATILE,
: 1261      2767  2  | Local temporary descriptor
: 1262      2768  2  TEMP      : VECTOR [10,BYTE], | Must hold up to 18 packed digits
: 1263      2769  2  TEMP_LEN : INITIAL (0), | Length of temporary desc
: 1264      2770  2  RES_DESC : BLOCK [12,BYTE] INITIAL (0) VOLATILE,
: 1265      2771  2  | Local temporary descriptor
: 1266      2772  2  RES      : VECTOR [23,BYTE], | Must hold up to 23 digits
: 1267      2773  2  | for double floating
: 1268      2774  2  TEMP_BUF : VECTOR [8,BYTE], | Needed for CH$TRANSLATE to reshuffle
: 1269      2775  2  STRING_BUF: REF VECTOR [8,BYTE], | Needed for DSC$K_DTYPE_NLO
: 1270      2776  2  | conversion
: 1271      2777  2  SIGN      : INITIAL (0), | Hold sign for DSC$K_DTYPE_NLO
: 1272      2778  2  WORD_TO_LONG : INITIAL (0), | Needed for CVTTL
: 1273      2779  2  STR_LEN   : WORD INITIAL (0), | Used by STR$COPY_R
: 1274      2780  2  LOOK_FOR_SIGN : INITIAL (0), | = 1 if data type requires sign
: 1275      2781  2  EXPONENT, | Scale of string to be converted
: 1276      2782  2  DIGITS    : INITIAL (0), | Number of digits in a COMP data item
: 1277      2783  2  CHECK_COMP : INITIAL (0), | TRUNC / NOTRUNC
: 1278      2784  2  COMP_SCALE : INITIAL (0), | TRUNC / NOTRUNC
: 1279      2785  2  PASS_RES  : INITIAL (0), | Flag for DISP_PARSE call
: 1280      2786  2  ITS_TEXT  : INITIAL (0), | Flag, if = 1 no need to call
: 1281      2787  2  | DISP_PARSE
: 1282      2788  2  LITERAL
: 1283      2789  2  F_SIZE    = 7, | Needed for call to COB$CNVOUT in DISP_CONVERT
: 1284      2790  2  D_SIZE    = 16, | Needed for call to COB$CNVOUT in DISP_CONVERT
: 1285      2791  2  OVERPUNCH_NEG_ZERO = %X'7D', | Representation of overpunch -0
: 1286      2792  2  LOW_OVERPUNCH_NEG_SIGN = %X'4A', | Representation of overpunch -1
: 1287      2793  2  HIGH_OVERPUNCH_NEG_SIGN = %X'52', | Representation of overpunch -9
: 1288      2794  2
: 1289      2795  2  !+
: 1290      2796  2  | Create local descriptors - TEMP_DESC and RES_DESC.
: 1291      2797  2  | STR$GET1_DX and STR$FREE1_DX are not used in this routine
: 1292      2798  2  | because we are dealing with CLASSES other than Dynamic
: 1293      2799  2  | -
: 1294      2800  2
: 1295      2801  2  TEMP_DESC [DSC$W_LENGTH] = .STRING [DSC$W_LENGTH] ;
: 1296      2802  2  TEMP_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_P ;
: 1297      2803  2  TEMP_DESC [DSC$B_CLASS] = DSC$K_CLASS_SD ;
: 1298      2804  2  TEMP_DESC [DSC$A_POINTER] = TEMP[0] ;
: 1299      2805  2  TEMP_LEN = .STRING[DSC$W_LENGTH] ;
: 1300      2806  2
: 1301      2807  2  RES_DESC [DSC$W_LENGTH] = .STRING [DSC$W_LENGTH] ;
: 1302      2808  2  RES_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NL ;
: 1303      2809  2  RES_DESC [DSC$B_CLASS] = DSC$K_CLASS_SD ;
: 1304      2810  2  RES_DESC [DSC$A_POINTER] = RES[0] ;
: 1305      2811  2
: 1306      2812  2  IF .STRING [DSC$B_CLASS] EQL DSC$K_CLASS_SD
: 1307      2813  2  THEN
: 1308      2814  2  BEGIN
: 1309      2815  2
: 1310      2816  2  TEMP_DESC [DSC$B_SCALE] = .STRING [DSC$B_SCALE] ;
```



```
1311 2817 3      TEMP_DESC [DSC$B_DIGITS] = .STRING [DSC$B_DIGITS] ;
1312 2818      RES_DESC [DSC$B_SCALE] = .STRING [DSC$B_SCALE] ;
1313 2819      RES_DESC [DSC$B_DIGITS] = .STRING [DSC$B_DIGITS] ;
1314 2820
1315 2821      END;
1316 2822
1317 2823      !+
1318 2824      ! Get EXPONENT, if class is DSC$K_CLASS_SD, to pass to DISP_PARSE
1319 2825      !-
1320 2826
1321 2827      EXPONENT = ( IF .STRING [DSC$B_CLASS] EQL DSC$K_CLASS_SD
1322 2828                  THEN .STRING [DSC$B_SCALE]
1323 2829                  ELSE 0 ) ;
1324 2830
1325 2831      !+
1326 2832      ! Select DATA TYPE of string to be converted and perform the
1327 2833      ! necessary conversions.
1328 2834      ! Object of conversion is to fold all data types to Text.
1329 2835
1330 2836      Values :
1331 2837
1332 2838      ! Although this routine converts all the various data types to TEXT,
1333 2839      ! it is necessary to remember what the original data types were.
1334 2840      ! This information is stored in LOOK_FOR_SIGN (with data types grouped
1335 2841      ! together when possible) to be used by routine DISP_PARSE in deciding
1336 2842      ! whether a plus sign, minus sign, or space is to be output.
1337 2843
1338 2844      LOOK_FOR_SIGN = 0      No sign insertion
1339 2845      LOOK_FOR_SIGN = 1      Sign is part of string (leading)
1340 2846      LOOK_FOR_SIGN = 2      Trailing sign
1341 2847      LOOK_FOR_SIGN = 3      Pos overpunch sign, COMP data types (word,
1342 2848                          ! longword, and quadword), and Packed data type
1343 2849      LOOK_FOR_SIGN = 4      '-' sign insertion
1344 2850                          ! DSC$K_DTYPE_NLO case where minus sign 'gets
1345 2851                          ! lost' in conversion.
1346 2852
1347 2853      PASS_RES = 0      Pass STRING to DISP_PARSE
1348 2854      PASS_RES = 1      Pass RES_DESC to DISP_PARSE
1349 2855
1350 2856      ITS_TEXT = 0      Call DISP_PARSE
1351 2857      ITS_TEXT = 1      No need to call DISP_PARSE
1352 2858
1353 2859      !-
1354 2860      CASE .STRING [DSC$B_DTYPE] FROM DSC$K_DTYPE_WU TO DSC$K_DTYPE_P OF
1355 2861      SET
1356 2862
1357 2863          [DSC$K_DTYPE_NU] :      ! Unsigned numeric
1358 2864
1359 2865              LOOK_FOR_SIGN = 0 ;
1360 2866
1361 2867          [DSC$K_DTYPE_NL] :      ! Left separate sign
1362 2868
1363 2869              LOOK_FOR_SIGN = 1 ;
1364 2870
1365 2871          [DSC$K_DTYPE_NR] :      ! Right separate sign
1366 2872
1367 2873      !+
1367 2873      2
```



```
: 1368      2874      2      ! EXPONENT is adjusted because of trailing sign that is
: 1369      2875      2      ! included in the string.
: 1370      2876      2      !-
: 1371      2877      2
: 1372      2878      2
: 1373      2879      2      BEGIN
: 1374      2880      2      IF .EXPONENT LSS 0
: 1375      2881      2      THEN
: 1376      2882      2          EXPONENT = .EXPONENT - 1 ;
: 1377      2883      2      LOOK_FOR_SIGN = 2 ;
: 1378      2884      2      END ;
: 1379      2885      2      [DSC$K_DTYPE_NRO] :      ! Right overpunch sign
: 1380      2886      2
: 1381      2887      2      BEGIN
: 1382      2888      2      CVTTP ( STRING[DSC$W_LENGTH], .STRING[DSC$A_POINTER],
: 1383      2889      2          LIB$AB_CVTTP_O, TEMP_LEN, TEMP );
: 1384      2890      2
: 1385      2891      2      CVTPS ( TEMP_LEN, TEMP, TEMP_LEN, RES );
: 1386      2892      2
: 1387      2893      2      LOOK_FOR_SIGN = 3 ;
: 1388      2894      2      PASS_RES = 1 ;
: 1389      2895      2      END ;
: 1390      2896      2
: 1391      2897      2      [DSC$K_DTYPE_NLO] :      ! Left overpunch sign
: 1392      2898      2
: 1393      2899      2      BEGIN
: 1394      2900      2
: 1395      2901      2      LOOK_FOR_SIGN = 3 ;
: 1396      2902      2
: 1397      2903      2      !+
: 1398      2904      2      ! CH$TRANSLATE loses '-' sign.
: 1399      2905      2      ! Read sign before performing CH$TRANSLATE. If a minus
: 1400      2906      2      ! sign was part of the original string, preserve it through
: 1401      2907      2      ! LOOK_FOR_SIGN. DISP_PARSE will insert the lost minus sign
: 1402      2908      2      ! in the final form of the string.
: 1403      2909      2      ! Note : a plus sign is not lost, plus sign is always given
: 1404      2910      2      ! by CH$TRANSLATE regardless of the original sign.
: 1405      2911      2      !-
: 1406      2912      2
: 1407      2913      2      STRING_BUF = .STRING[DSC$A_POINTER] ;
: 1408      2914      2      SIGN = .STRING_BUF[0] ;
: 1409      2915      2      IF .SIGN GEQ LOW_OVERPUNCH_NEG_SIGN AND      ! If between -1
: 1410      2916      2          .SIGN LEQ HIGH_OVERPUNCH_NEG_SIGN      ! and -9 then
: 1411      2917      2      THEN      ! preserve neg sign
: 1412      2918      2          LOOK_FOR_SIGN = 4
: 1413      2919      2      ELSE
: 1414      2920      2          IF .SIGN EQL OVERPUNCH_NEG_ZERO      ! If -0 then
: 1415      2921      2      THEN      ! preserve neg sign
: 1416      2922      2          LOOK_FOR_SIGN = 4 ;
: 1417      2923      2
: 1418      2924      2
: 1419      2925      2      CH$TRANSLATE ( LIB$AB_CVT_O_U, .STRING[DSC$W_LENGTH],
: 1420      2926      2          .STRING[DSC$A_POINTER], 0,
: 1421      2927      2          .STRING[DSC$W_LENGTH], TEMP_BUF[0] );
: 1422      2928      2
: 1423      2929      2      CVTTP ( STRING[DSC$W_LENGTH], TEMP_BUF[0],
: 1424      2930      2          LIB$AB_CVTTP_U, TEMP_LEN, TEMP );
```

```
1425 2931
1426 2932      CVTPS ( TEMP_LEN, TEMP, TEMP_LEN, RES );
1427 2933
1428 2934      PASS_RES = 1 ;
1429 2935      END ;
1430 2936
1431 2937 [DSC$K_DTYPE_P] :      ! Packed decimal
1432 2938
1433 2939      BEGIN
1434 2940      CVTPS ( STRING[DSC$W_LENGTH], .STRING[DSC$A_POINTER],
1435 2941              STRING[DSC$W_LENGTH], RES );
1436 2942
1437 2943      LOOK_FOR_SIGN = 3 ;
1438 2944      PASS_RES = 1 ;
1439 2945      CHECK_COMP = 2 ;
1440 2946      END ;
1441 2947
1442 2948 [DSC$K_DTYPE_W, DSC$K_DTYPE_WU] :      ! Signed and unsigned word
1443 2949
1444 2950 *** NOTE: For COMP data items (WORD, LONGWORD, QUADWORD), VAX-11 COBOL
1445 2951           is passing an SD decriptor for both the S and SD class.
1446 2952
1447 2953      BEGIN
1448 2954
1449 2955      !+
1450 2956      Although 4 is the maximum number of digits in a VAX-11
1451 2957      COBOL Word Integer, a length of 9 is used because conversion
1452 2958      is actually from Longword to Packed. Need the number of
1453 2959      digits possible in Longword.
1454 2960      Suppression of leading zeros will be necessary (done in
1455 2961      DISP_PARSE).
1456 2962      -
1457 2963
1458 2964      TEMP_LEN = 9 ;
1459 2965      RES DESC [DSC$W_LENGTH] = 9 ;
1460 2966      WORD_TO_LONG = .BLOCK[.STRING[DSC$A_POINTER],0,0,16,1;,.BYTE] ;
1461 2967
1462 2968      CVTLP ( WORD_TO_LONG , TEMP_LEN, TEMP ) ;
1463 2969      CVTPS ( TEMP_LEN, TEMP, TEMP_LEN, RES );
1464 2970
1465 2971      LOOK_FOR_SIGN = 3 ;
1466 2972      PASS_RES = 1 ;
1467 2973      !+
1468 2974      Read number digits in COMP data item to pass to DISP_PARSE.
1469 2975      .STRING [DSC$W_LENGTH] is always 2 for WORDs.
1470 2976      Number of digits is between 1 and 4 for WORDs.
1471 2977      If DIS$K_CLASS S -> During the conversion process the WORD
1472 2978      data type was first converted to a LONGWORD. This introduced
1473 2979      five 'extra' preceding zeroes which will be disgarded by
1474 2980      DISP_PARSE.
1475 2981      -
1476 2982      CHECK_COMP = 1 ;
1477 2983      IF .STRING[DSC$B_CLASS] EQL DSC$K_CLASS_SD
1478 2984      THEN
1479 2985          BEGIN
1480 2986              DIGITS = .STRING[DSC$B_DIGITS] ;
1481 2987              COMP_SCALE = .STRING[DSC$B_SCALE] ;
```



```
1482      2988      4
1483      2989
1484      2990
1485      2991
1486      2992
1487      2993
1488      2994
1489      2995
1490      2996
1491      2997
1492      2998
1493      2999
1494      3000
1495      3001
1496      3002
1497      3003
1498      3004
1499      3005
1500      3006
1501      3007
1502      3008
1503      3009
1504      3010
1505      3011
1506      3012
1507      3013
1508      3014
1509      3015
1510      3016
1511      3017
1512      3018
1513      3019
1514      3020
1515      3021
1516      3022
1517      3023
1518      3024
1519      3025
1520      3026
1521      3027
1522      3028
1523      3029
1524      3030
1525      3031
1526      3032
1527      3033
1528      3034
1529      3035
1530      3036
1531      3037
1532      3038
1533      3039
1534      3040
1535      3041
1536      3042
1537      3043
1538      3044

      END
    ELSE
      DIGITS = -1 ;
    END ;

[DSC$K_DTYPE_L, DSC$K_DTYPE_LU] :    ! Signed and unsigned longword
BEGIN
  +
  9 is the maximum number of digits in a VAX-11 COBOL
  Longword Integer.
  Suppression of leading zeros will be necessary if length
  of input string is less than 9 (done in DISP_PARSE).
  -

  TEMP_LEN = 9 ;
  RES_DESC [DSC$W_LENGTH] = 9 ;

  CVTLP ( .STRING[DSC$A_POINTER], TEMP_LEN, TEMP ) ;
  CVTPS ( TEMP_LEN, TEMP, TEMP_LEN, RES );

  LOOK_FOR_SIGN = 3 ;
  PASS_RES = 1 ;
  +
  Read number digits in COMP data item to pass to DISP_PARSE.
  .STRING [DSC$W_LENGTH] is always 4 for LONGWORDS.
  Number of digits is between 5 and 9 for LONGWORDS.
  -

  CHECK_COMP = 1 ;
  IF .STRING[DSC$B_CLASS] EQL DSC$K_CLASS_SD
  THEN
    BEGIN
      DIGITS = .STRING[DSC$B_DIGITS] ;
      COMP_SCALE = .STRING[DSC$B_SCALE] ;
    END
  ELSE
    DIGITS = 0 ;
  END ;

[DSC$K_DTYPE_QU, DSC$K_DTYPE_Q] :    ! Signed and unsigned quadword
BEGIN
  +
  18 is the maximum number of digits in a VAX-11 COBOL
  Quadword Integer.
  Suppression of leading zeros will be necessary if length
  of input string is less than 18 (done in DISP_PARSE).
  -

  TEMP_LEN = 18 ;
  RES_DESC [DSC$W_LENGTH] = 18 ;
  (COB$CVTQP R9 (0, .STRING[DSC$A_POINTER], .TEMP_LEN, TEMP)) ;
  CVTPS ( TEMP_LEN, TEMP, TEMP_LEN, RES );
```

```
1539 3045 LOOK_FOR_SIGN = 3 ;
1540 3046 PASS_RES = 1 ;
1541 3047
1542 3048 +
1543 3049 Read number digits in COMP data item to pass to DISP_PARSE.
1544 3050 .STRING [DSC$W_LENGTH] is always 8 for QUADWORDS.
1545 3051 Number of digits is between 10 and 18 for QUADWORDS.
1546 3052
1547 3053 CHECK_COMP = 1 ;
1548 3054 IF .STRING[DSC$B_CLASS] EQL DSC$K_CLASS_SD
1549 3055 THEN
1550 3056 BEGIN
1551 3057 DIGITS = .STRING[DSC$B_DIGITS] ;
1552 3058 COMP_SCALE = .STRING[DSC$B_SCALE] ;
1553 3059 END
1554 3060 ELSE
1555 3061 DIGITS = 0 ;
1556 3062
1557 3063 END ;
1558 3064 [DSC$K_DTYPE_F] : ! Floating point
1559 3065
1560 3066 BEGIN
1561 3067
1562 3068 +
1563 3069 14 is the length of the E-Notation format that is used
1564 3070 for the Floating Point data type.
1565 3071 ( E Notation representation of -1 is -0.1000000E+01 )
1566 3072 STR$COPY_R is done here because DISP_PARSE will not be called
1567 3073
1568 3074
1569 3075 RES_DESC [DSC$W_LENGTH] = 14 ;
1570 3076 STR_LEN = 14 ;
1571 3077 IF NOT ( COB$CNVOUT (.STRING[DSC$A_POINTER], RES_DESC, F_SIZE))
1572 3078 THEN
1573 3079 LIB$STOP (COB$ERRDURDIS) ;
1574 3080 IF NOT ( STR$COPY_R ( .ANS_STRING, STR_LEN,
1575 3081 .RES_DESC [DSC$A_POINTER] ) )
1576 3082 THEN
1577 3083 LIB$STOP (COB$ERRDURDIS) ;
1578 3084
1579 3085 +
1580 3086 Change the result 0.1110000E+03, to 1.110000E+02
1581 3087
1582 3088 ADJUST_FL_PT ;
1583 3089 ITS_TEXT = 1 ;
1584 3090 END ;
1585 3091 [DSC$K_DTYPE_D] : ! Double floating point
1586 3092
1587 3093 BEGIN
1588 3094
1589 3095 +
1590 3096 23 is the length of the E-Notation format that is used
1591 3097 for the Double Floating Point data type.
1592 3098 ( E Not representation of -1 is -0.1000000000000000E+01 )
1593 3099 STR$COPY_R is done here because DISP_PARSE will not be called
1594 3100
1595 3101
```



```

1596 3102 3 RES_DESC [DSC$W_LENGTH] = 23 ;
1597 3103 3 STR_LEN = 23 ;
1598 3104 3 IF NOT ( COB$CNVOUT ( .STRING[DSC$A_POINTER], RES_DESC, D_SIZE))
1599 3105 3 THEN
1600 3106 3     LIB$STOP (COB$ERRDURDIS) ;
1601 3107 3 IF NOT ( STR$COPY_R ( .ANS_STRING, STR_LEN,
1602 3108 3     .RES_DESC [DSC$A_POINTER] ) )
1603 3109 3 THEN
1604 3110 3     LIB$STOP (COB$ERRDURDIS) ;
1605 3111 3     + Change the result 0.1234567890123456E-07 to
1606 3112 3     1.234567890123456E-08 to
1607 3113 3     -
1608 3114 3     ADJUST_FL_PT ;
1609 3115 3     ITS_TEXT = 1 ;
1610 3116 3     END ;
1611 3117 3
1612 3118 3 [DSC$K_DTYPE_T] : ! Text
1613 3119 3
1614 3120 3 BEGIN
1615 3121 3
1616 3122 3     +
1617 3123 3     User requested conversion of a TEXT string. Empty request,
1618 3124 3     no harm done - don't call DISP_PARSE,
1619 3125 3     STR$COPY_R is done here.
1620 3126 3     -
1621 3127 3
1622 3128 3 STR_LEN = .STRING [DSC$W_LENGTH] ;
1623 3129 3 IF NOT ( STR$COPY_R ( .ANS_STRING, STR_LEN,
1624 3130 3     .STRING [DSC$A_POINTER] ) )
1625 3131 3 THEN
1626 3132 3     LIB$STOP (COB$ERRDURDIS) ;
1627 3133 3     ITS_TEXT = 1 ;
1628 3134 3     END ;
1629 3135 3
1630 3136 3
1631 3137 3 [INRANGE, OUTRANGE] :
1632 3138 3     LIB$STOP(COB$INVARG) ;
1633 3139 3 TES ;
1634 3140 3
1635 3141 3
1636 3142 3 +
1637 3143 3 Call routine to parse the converted string and put it in acceptable
1638 3144 3 form for the call to $PUT in COMMON_SCREEN.
1639 3145 3 DISP_PARSE will
1640 3146 3     insert sign
1641 3147 3     insert decimal point or comma
1642 3148 3     suppress leading zeroes
1643 3149 3     copy final form of input string to ANS_STRING
1644 3150 3     -
1645 3151 3
1646 3152 3 IF .ITS_TEXT EQL 0
1647 3153 3 THEN
1648 3154 3     BEGIN
1649 3155 3         LOCAL
1650 3156 3         YES_ZERO : INITIAL (0) ;
1651 3157 3         IF (.STRING[DSC$B_CLASS] EQL DSC$K_CLASS_S)
1652 3158 3
```

```

: 1653      3159      3
: 1654      3160      3
: 1655      3161      3
: 1656      3162      4
: 1657      3163      4
: 1658      3164      4
: 1659      3165      4
: 1660      3166      4
: 1661      3167      3
: 1662      3168      3
: 1663      3169      4
: 1664      3170      4
: 1665      3171      3
: 1666      3172      3
: 1667      3173      3
: 1668      3174      3
: 1669      3175      2
: 1670      3176      2
: 1671      3177      1

      THEN
      YES_ZERO = 1
    ELSE
      BEGIN
      YES_ZERO = .STRING[DSC$B_DIGITS] + .STRING[DSC$B_SCALE] ;
      IF .YES_ZERO NEQ 0
      THEN
      YES_ZERO = 1 ;
      END ;
    DISP_PARSE (( IF .PASS_RES
      THEN RES_DESC
      ELSE .STRING ) ,
      .FLAGS, .LOOK_FOR_SIGN, .EXPONENT, .DIGITS, .ANS_STRING,
      .YES_ZERO, .COMP_SCALE, .CHECK_COMP ) ;
    END ;
  END ;
! End DISP_CONVERT
```

```

      00000000 0057E .BLKB 2
      00000000 00580 P.AAR: .LONG 0
      00000000 00584 P.AAS: .LONG 0
      .EXTRN LIB$AB_CVTTP_O, LIB$AB_CVTTP_U
      .EXTRN LIB$AB_CVT_O_U

      OFFC 00000 DISP_CONVERT:
      .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
      MOVAB -104(SP), SP
      MOVCS #4, P.AAR, #0, #12, TEMP_DESC
      CLRL TEMP_LEN
      MOVCS #4, P.AAS, #0, #12, RES_DESC
      CLRL SIGN
      CLRL STR_LEN
      CLRL DIGITS
      CLRL COMP_SCALE
      CLRL ITS_TEXT
      MOVL STRING, R10
      MOVW (R10), TEMP_DESC
      MOVW #21, TEMP_DESC+2
      MOVW #9, TEMP_DESC+3
      MOVAB TEMP, TEMP_DESC+4
      MOVZWL (R10), TEMP_LEN
      MOVW (R10), RES_DESC
      MOVW #16, RES_DESC+2
      MOVW #9, RES_DESC+3
      MOVAB RES, RES_DESC+4
      CLRL (SP)
      CMPB 3(R10), #9
      BNEQ 1$
      INCL (SP)
      MOVW 8(R10), TEMP_DESC+8

      2703
      2766
      2770
      2801
      2802
      2803
      2804
      2805
      2807
      2808
      2809
      2810
      2812
      2816
```


0026	0107	00F8	09	AA	90	0005E	MOVB	9(R10), TEMP_DESC+9	2817
0149	0107	00F8	08	AA	90	00063	MOVB	8(R10), RES_DESC+8	2818
033E	0026	0026	09	AA	90	00068	MOVB	9(R10), RES_DESC+9	2819
0040	0067	003A		6E	E9	0006D	BLBC	(SP), 2\$	2827
	00B7	0026	08	AA	98	00070	CVTBL	8(R10), EXPONENT	2828
				03	11	00075	BRB	3\$	
			1C	AE	D4	00077	CLRL	EXPONENT	2827
			02	AA	8F	0007A	CASEB	2(R10), #3, #18	2860
				00CC		0007F	.WORD	20\$-4\$,-	
				00CC		00087		22\$-4\$,-	
				0243		0008F		23\$-4\$,-	
				0035		00097		5\$-4\$,-	
				004E		0009F		20\$-4\$,-	
								22\$-4\$,-	
								23\$-4\$,-	
								28\$-4\$,-	
								45\$-4\$,-	
								5\$-4\$,-	
								5\$-4\$,-	
								63\$-4\$,-	
								6\$-4\$,-	
								7\$-4\$,-	
								12\$-4\$,-	
								8\$-4\$,-	
								11\$-4\$,-	
								5\$-4\$,-	
								18\$-4\$	
								#COB\$ INVARG	3139
								#1, LIB\$STOP	
								10\$	
								LOOK_FOR_SIGN	2865
								17\$	
								#1, LOOK_FOR_SIGN	2869
								17\$	
								EXPONENT	2879
								9\$	
								EXPONENT	2881
								#2, LOOK_FOR_SIGN	2882
								19\$	2860
								(R10), a4(R10), LIB\$AB_CVTTP_O, TEMP_LEN, -	2888
								TEMP	
								TEMP_LEN, TEMP, TEMP_LEN, RES	2891
								#3, LOOK_FOR_SIGN	2893
								16\$	2894
								#3, LOOK FOR SIGN	2901
								4(R10), STRING BUF	2913
								(STRING BUF), SIGN	2914
								SIGN, #74	2915
								13\$	
								SIGN, #82	2916
								14\$	
								SIGN, #125	2920
								15\$	
								#4, LOOK FOR SIGN	2922
								(R10), a4(R10), #0, LIB\$AB_CVT_O_U, (R10), -	2927
								TEMP BUF	
								(R10), TEMP_BUF, LIB\$AB_CVTTP_U, TEMP_LEN, -	2929

2C	AE	5B	50	AE	50	AE	00127	TEMP	TEMP_LEN, TEMP, TEMP_LEN, RES	2932
			08	AE		5B	08 00129	CVTPS	TEMP_LEN, TEMP, TEMP_LEN, RES	2934
						01	DO 00130	MOVL	#1, PASS_RES	2860
2C	AE	6A	04	BA		3F	11 00134	BRB	21\$	2941
			18	AE		6A	08 00136	CVTPS	(R10), @4(R10), (R10), RES	2943
			08	AE		03	DO 0013D	MOVL	#3, LOOK_FOR_SIGN	2944
			10	AE		01	DO 00141	MOVL	#1, PASS_RES	2945
						02	DO 00145	MOVL	#2, CHECK_COMP	2860
				5B		7A	11 00149	BRB	27\$	2964
			44	AE		09	DO 0014B	MOVL	#9, TEMP_LEN	2965
				55		09	BO 0014E	MOVW	#9, RES_DESC	2966
2C	AE	50	AE	5B	04	BA	32 00152	CVTL	@4(R10), WORD_TO_LONG	2968
			50	AE		55	F9 00156	CVTL	WORD_TO_LONG, TEMP_LEN, TEMP	2969
			18	AE		5B	08 0015B	CVTPS	TEMP_LEN, TEMP, TEMP_LEN, RES	2971
			08	AE		03	DO 00162	MOVL	#3, LOOK_FOR_SIGN	2972
			10	AE		01	DO 00166	MOVL	#1, PASS_RES	2982
				45		01	DO 0016A	MOVL	#1, CHECK_COMP	2983
			14	AE		6E	E8 0016E	BLBS	(SP), 25\$	2990
						01	CE 00171	MNEGL	#1, DIGITS	2860
				5B		4E	11 00175	BRB	27\$	3004
			44	AE		09	DO 00177	MOVL	#9, TEMP_LEN	3005
		50	AE	5B	04	09	BO 0017A	MOVW	#9, RES_DESC	3007
						BA	F9 0017E	CVTL	@4(R10), TEMP_LEN, TEMP	3008
				5B		1A	11 00184	BRB	24\$	3040
			44	AE		12	DO 00186	MOVL	#18, TEMP_LEN	3041
				59		12	BO 00189	MOVW	#18, RES_DESC	3042
				58		AE	9E 0018D	MOVAB	TEMP, R9	
				57		5B	DO 00191	MOVL	TEMP_LEN, R8	
					04	AA	DO 00194	MOVL	4(R10), R7	
						56	D4 00198	CLRL	R6	
					00000000G	00	16 0019A	JSB	COB\$CVTQ_R9	
2C	AE	5B	50	AE		5B	08 001A0	CVTPS	TEMP_LEN, TEMP, TEMP_LEN, RES	3043
			18	AE		03	DO 001A7	MOVL	#3, LOOK_FOR_SIGN	3045
			08	AE		01	DO 001AB	MOVL	#1, PASS_RES	3046
			10	AE		01	DO 001AF	MOVL	#1, CHECK_COMP	3052
				OC		6E	E9 001B3	BLBC	(SP), 26\$	3053
			14	AE		AA	9A 001B6	MOVZBL	9(R10), DIGITS	3056
			OC	AE		08	AA 98 001BB	CVTBL	8(R10), COMP_SCALE	3057
						03	11 001C0	BRB	27\$	3053
						AE	D4 001C2	CLRL	DIGITS	3060
						021D	31 001C5	BRW	65\$	2860
			44	AE		0E	BO 001C8	MOVW	#14, RES_DESC	3075
			20	AE		0E	BO 001CC	MOVW	#14, STR_LEN	3076
						07	DD 001D0	PUSHL	#7	3077
						AE	9F 001D2	PUSHAB	RES_DESC	
						AA	DD 001D5	PUSHL	4(R10)	
						03	FB 001D8	CALLS	#3, COB\$CNVOUT	
						50	E8 001DF	BLBS	R0, 29\$	
						8F	DD 001E2	PUSHL	#COB\$ ERRDURDIS	3079
						01	FB 001E8	CALLS	#1, LIB\$STOP	
						AE	DD 001EF	PUSHL	RES_DESC+4	3081
						AE	9F 001F2	PUSHAB	STR_LEN	3080
						AC	DO 001F5	MOVL	ANS_STRING, R4	
						54	DD 001F9	PUSHL	R4	
						03	FB 001FB	CALLS	#3, STR\$COPY_R	
						50	E8 00202	BLBS	R0, 30\$	
						8F	DD 00205	PUSHL	#COB\$ ERRDURDIS	3083


```
00000000G 00 01 FB 0020B CALLS #1, LIB$STOP
50 04 A4 D0 00212 30$: MOVL 4(R4), ANS_BUF
0A 02 AA 91 00216 CMPB 2(R10), #10
56 0E 12 0021A BNEQ 31$
51 0B D0 0021C MOVL #11, E_SIGN
52 0D D0 0021F MOVL #13, E_ONES
58 0C D0 00222 MOVL #12, E_TENS
0C D0 00225 MOVL #12, SHIFT_ALL
0C 11 00228 BRB 32$
56 14 D0 0022A 31$: MOVL #20, E_SIGN
51 16 D0 0022D MOVL #22, E_ONES
52 15 D0 00230 MOVL #21, E_TENS
58 15 D0 00233 MOVL #21, SHIFT_ALL
04 08 AC 06 E1 00236 32$: BBC #6, FLAGS, 33$
02 A0 2C 90 0023B MOVB #44, 2(ANS_BUF)
51 50 C0 0023F 33$: ADDL2 ANS_BUF, RT
30 61 91 00242 CMPB (R1), #48
37 37 12 00245 BNEQ 38$
30 6240 91 00247 CMPB (E_TENS)[ANS_BUF], #48
31 31 12 0024B BNEQ 38$
0C 58 D1 0024D CMPL SHIFT_ALL, #12
55 05 12 00250 BNEQ 34$
55 09 D0 00252 MOVL #9, SEARCH
53 03 11 00255 BRB 35$
53 12 D0 00257 34$: MOVL #18, SEARCH
09 02 D0 0025A 35$: MOVL #2, P
30 09 11 0025D BRB 37$
6340 91 0025F 36$: CMPB (P)[ANS_BUF], #48
03 13 00263 BEQL 37$
57 01 D0 00265 MOVL #1, CHANGE
53 55 F3 00268 37$: AOBLEQ SEARCH, P, 36$
01 57 D1 0026C CMPL CHANGE, #1
6640 33 12 0026F BNEQ 42$
6240 2D 90 00271 MOVB #45, (E_SIGN)[ANS_BUF]
61 30 90 00275 MOVB #48, (E_TENS)[ANS_BUF]
26 31 90 00279 MOVB #49, (RT)
2B 6640 91 0027C BRB 42$
11 6640 91 0027E 38$: CMPB (E_SIGN)[ANS_BUF], #42
30 61 12 00282 BNEQ 40$
04 61 91 00284 CMPB (R1), #48
61 04 13 00287 BEQL 39$
17 61 97 00289 DECB (R1)
39 39 90 0028D 39$: BRB 42$
6240 97 00290 MOVB #57, (R1)
0F 11 00293 DECB (E_TENS)[ANS_BUF]
39 61 91 00295 40$: BRB 42$
04 04 13 00298 CMPB (R1), #57
61 61 96 0029A BEQL 41$
06 11 0029C INCB (R1)
61 30 90 0029E 41$: BRB 42$
6240 96 002A1 MOVB #48, (R1)
51 02 A0 90 002A4 42$: INCB (E_TENS)[ANS_BUF]
03 A0 90 002A8 MOVB 2(ANS_BUF), TEMP
51 51 90 002AD MOVB 3(ANS_BUF), 2(ANS_BUF)
06 11 002B3 CLRL TEMP, 3(ANS_BUF)
BRB X
44$
```


F6	6140	01	A140	90	002B5	43\$:	MOVB	1(X)[ANS_BUF], (X)[ANS_BUF]	
	51		58	F3	002BB	44\$:	AOBLEQ	SHIFT_ALC, X, 43\$	
			00F7	31	002BF		BRW	62\$	
	44	AE	17	B0	002C2	45\$:	MOVW	#23, RES_DESC	3102
	20	AE	17	B0	002C6		MOVW	#23, STR_LEN	3103
			10	DD	002CA		PUSHL	#16	3104
		48	AE	9F	002CC		PUSHAB	RES_DESC	
		04	AA	DD	002CF		PUSHL	4(RT0)	
00000000G	00		03	FB	002D2		CALLS	#3, COB\$CNVOUT	
	0D		50	E8	002D9		BLBS	R0, 46\$	
00000000G	00	00000000G	8F	DD	002DC		PUSHL	#COB\$ERRDURDIS	3106
			01	FB	002E2		CALLS	#1, LIB\$STOP	
		48	AE	DD	002E9	46\$:	PUSHL	RES_DESC+4	3108
		24	AE	9F	002EC		PUSHAB	STR_LEN	3107
	54	0C	AC	DD	002EF		MOVL	ANS_STRING, R4	
			54	DD	002F3		PUSHL	R4	
00000000G	00		03	FB	002F5		CALLS	#3, STR\$COPY_R	
	0D		50	E8	002FC		BLBS	R0, 47\$	
00000000G	00	00000000G	8F	DD	002FF		PUSHL	#COB\$ERRDURDIS	3110
			01	FB	00305		CALLS	#1, LIB\$STOP	
	50		A4	DD	0030C	47\$:	MOVL	4(R4), ANS_BUF	
	0A	02	AA	91	00310		CMPB	2(R10), #10	
			0E	12	00314		BNEQ	48\$	
	56		0B	DD	00316		MOVL	#11, E_SIGN	
	51		0D	DD	00319		MOVL	#13, E_ONES	
	52		0C	DD	0031C		MOVL	#12, E_TENS	
	58		0C	DD	0031F		MOVL	#12, SHIFT_ALL	
			0C	11	00322		BRB	49\$	
	56		14	DD	00324	48\$:	MOVL	#20, E_SIGN	
	51		16	DD	00327		MOVL	#22, E_ONES	
	52		15	DD	0032A		MOVL	#21, E_TENS	
	58		15	DD	0032D		MOVL	#21, SHIFT_ALL	
04	08		06	E1	00330	49\$:	BBC	#6, FLAGS, -50\$	
	02		2C	90	00335		MOVB	#44, 2(ANS_BUF)	
			50	C0	00339	50\$:	ADDL2	ANS_BUF, RT	
	51		61	91	0033C		CMPB	(RT), #48	
	30		37	12	0033F		BNEQ	55\$	
		6240	91	00341		CMPB	(E_TENS)[ANS_BUF], #48		
	30		31	12	00345		BNEQ	55\$	
	0C		58	D1	00347		CMPL	SHIFT_ALL, #12	
			05	12	0034A		BNEQ	51\$	
	55		09	DD	0034C		MOVL	#9, SEARCH	
			03	11	0034F		BRB	52\$	
	55		12	DD	00351	51\$:	MOVL	#18, SEARCH	
	53		02	DD	00354	52\$:	MOVL	#2, P	
			09	11	00357		BRB	54\$	
	30		6340	91	00359	53\$:	CMPB	(P)[ANS_BUF], #48	
			03	13	0035D		BEQL	54\$	
	57		01	DD	0035F		MOVL	#1, CHANGE	
F3	53		55	F3	00362	54\$:	AOBLEQ	SEARCH, P, 53\$	
	01		57	D1	00366		CMPL	CHANGE, #1	
			33	12	00369		BNEQ	59\$	
	6640		2D	90	0036B		MOVB	#45, (E_SIGN)[ANS_BUF]	
	6240		30	90	0036F		MOVB	#48, (E_TENS)[ANS_BUF]	
	61		31	90	00373		MOVB	#49, (RT)	
			26	11	00376		BRB	59\$	
	2B		6640	91	00378	55\$:	CMPB	(E_SIGN)[ANS_BUF], #43	

			11	12	0037C	BNEQ	57\$		
	30		61	91	0037E	CMPB	(R1), #48		
			04	13	00381	BEQL	56\$		
			61	97	00383	DECB	(R1)		
			17	11	00385	BRB	59\$		
	61		39	90	00387	56\$: MOVB	#57, (R1)		
		6240	97	0038A	DECB	(E TENS)[ANS_BUF]			
		0F	11	0038D	BRB	59\$			
	39		61	91	0038F	57\$: CMPB	(R1), #57		
			04	13	00392	BEQL	58\$		
			61	96	00394	INCB	(R1)		
			06	11	00396	BRB	59\$		
	61		30	90	00398	58\$: MOVB	#48, (R1)		
		6240	96	0039B	INCB	(E TENS)[ANS_BUF]			
	51	02	A0	90	0039E	59\$: MOVB	2(ANS_BUF), TEMP		
02	A0	03	A0	90	003A2	MOVB	3(ANS_BUF), 2(ANS_BUF)		
03	A0		51	90	003A7	MOVB	TEMP, 3(ANS_BUF)		
			51	D4	003AB	CLRL	X		
			06	11	003AD	BRB	61\$		
F6	6140	01	A140	90	003AF	60\$: MOVB	1(X)[ANS_BUF], (X)[ANS_BUF]		
	51		58	F3	003B5	61\$: AOBLEQ	SHIFT_ALC, X, 60\$		
			64	B7	003B9	62\$: DECB	(R4)		
			24	11	003BB	BRB	64\$		3116
	20	AE	6A	B0	003BD	63\$: MOVW	(R10), STR_LEN		3129
		04	AA	DD	003C1	PUSHL	4(R10)		3131
		24	AE	9F	003C4	PUSHAB	STR_LEN		3130
		0C	AC	DD	003C7	PUSHL	ANS_STRING		
00000000G	00		03	FB	003CA	CALLS	#3, STR\$COPY_R		
	0D		50	E8	003D1	BLBS	R0, 64\$		
00000000G	00	00000000G	8F	DD	003D4	PUSHL	#COB\$ ERRDURDIS		3133
	04	AE	01	FB	003DA	CALLS	#1, LIB\$STOP		
			01	D0	003E1	64\$: MOVL	#1, ITS_TEXT		3134
		04	AE	D5	003E5	65\$: TSTL	ITS_TEXT		3152
			3E	12	003E8	BNEQ	69\$		
			50	D4	003EA	CLRL	YES_ZERO		3154
	01	03	AA	91	003EC	CMPB	3(R10), #1		3158
			0D	13	003F0	BEQL	66\$		
	50	09	AA	9A	003F2	MOVZBL	9(R10), YES_ZERO		3163
	51	08	AA	98	003F6	CVTBL	8(R10), R1		
	50		51	C0	003FA	ADDL2	R1, YES_ZERO		
			03	13	003FD	BEQL	67\$		3164
	50		01	D0	003FF	66\$: MOVL	#1, YES_ZERO		3166
		10	AE	DD	00402	67\$: PUSHL	CHECK_COMP		3174
		10	AE	DD	00405	PUSHL	COMP_SCALE		
			50	DD	00408	PUSHL	YES_ZERO		
		0C	AC	DD	0040A	PUSHL	ANS_STRING		3173
		24	AE	DD	0040D	PUSHL	DIGITS		
		30	AE	DD	00410	PUSHL	EXPONENT		
		30	AE	DD	00413	PUSHL	LOOK_FOR_SIGN		
		08	AC	DD	00416	PUSHL	FLAGS		
	04		28	AE	E9	00419	BLBC	PASS_RES, 68\$	3169
	5A		64	AE	9E	0041D	MOVAB	RES_DESC, R10	
			5A	DD	00421	68\$: PUSHL	R10		3171
0000V	CF		09	FB	00423	CALLS	#9, DISP_PARSE		3169
			04	00428	69\$: RET				3177

; Routine Size: 1065 bytes, Routine Base: _COB\$CODE + 0588

COB\$DISPLAY
1-015

VAX-11 COBOL DISPLAY statement

D 7
16-Sep-1984 00:02:31
14-Sep-1984 12:10:42

VAX-11 Bliss-32 V4.0-742
[COBRTL.SRC]COBDISPLA.B32;1

Page 48
(11)


```
: 1673 3178 1 ROUTINE DISP_PARSE (
: 1674 3179 1     STRING      : REF BLOCK [12, BYTE],
: 1675 3180 1     FLAGS,
: 1676 3181 1     LOOK_FOR_SIGN,
: 1677 3182 1     EXPONENT,
: 1678 3183 1     DIGITS,
: 1679 3184 1     ANS_STRING : REF BLOCK [12, BYTE],
: 1680 3185 1
: 1681 3186 1     YES_ZERO,
: 1682 3187 1     COMP_SCALE,
: 1683 3188 1     CHECK_COMP
: 1684 3189 1
: 1685 3190 1
: 1686 3191 1     ) : NOVALUE =
: 1687 3192 1
: 1688 3193 1
: 1689 3194 1 ++
: 1690 3195 1     FUNCTIONAL DESCRIPTION:
: 1691 3196 1         Put string in final form for output -
: 1692 3197 1             insert sign
: 1693 3198 1             insert decimal point or comma
: 1694 3199 1             suppress leading zeroes
: 1695 3200 1             copy final form of input string to ANS_STRING
: 1696 3201 1
: 1697 3202 1
: 1698 3203 1     FORMAL PARAMETERS:
: 1699 3204 1
: 1700 3205 1         STRING.rt.dx      address of input string descriptor
: 1701 3206 1
: 1702 3207 1         FLAGS.rlu.v       screen enhancement flag
: 1703 3208 1
: 1704 3209 1             bit 0 - bold
: 1705 3210 1             bit 1 - reverse
: 1706 3211 1             bit 2 - blinking
: 1707 3212 1             bit 3 - underline
: 1708 3213 1             bit 4 - bell
: 1709 3214 1             bit 5 - conversion
: 1710 3215 1             bit 6 - decimal point is comma
: 1711 3216 1             bit 7 - 0 print sign, 1 do not print sign
: 1712 3217 1             bit 11 - 0 for VAX COBOL, 1 for VAX RPG
: 1713 3218 1
: 1714 3219 1         LOOK_FOR_SIGN.rlu.v  flag set in DISP_CONVERT to aid in sign insertion
: 1715 3220 1             and parameter list for call to DISP_PARSE.
: 1716 3221 1
: 1717 3222 1             0  No sign insertion
: 1718 3223 1             1  Sign is part of string (leading)
: 1719 3224 1             2  Trailing sign
: 1720 3225 1             3  Pos overpunch sign, COMP data types (word,
: 1721 3226 1             longword, and quadword) and packed data type
: 1722 3227 1             4  minus sign must be inserted (DSC$DTYPE_NLO
: 1723 3228 1             case)
: 1724 3229 1
: 1725 3230 1         EXPONENT.rlu.v      Decimal exponent - signed power of ten from DSC$B_SCALE
: 1726 3231 1             used to convert internal to external form
: 1727 3232 1
: 1728 3233 1         DIGITS.rl.v              Number of digits in a VAX-11 COBOL COMP data item,
: 1729 3234 1             Word, Longword and Quadword.
```

```
: 1730      3235 1 |
: 1731      3236 1 |      ANS_STRING.wt.dx address of descriptor to hold final form of
: 1732      3237 1 |                      string to be displayed on specified device
: 1733      3238 1 |
: 1734      3239 1 | IMPLICIT INPUTS:
: 1735      3240 1 |
: 1736      3241 1 |     NONE
: 1737      3242 1 |
: 1738      3243 1 | IMPLICIT OUTPUTS:
: 1739      3244 1 |
: 1740      3245 1 |     Status updated for STR$COPY_R
: 1741      3246 1 |
: 1742      3247 1 | ROUTINE VALUE:
: 1743      3248 1 |
: 1744      3249 1 |     NONE
: 1745      3250 1 |
: 1746      3251 1 | SIDE EFFECTS:
: 1747      3252 1 |
: 1748      3253 1 |     Copy the final form of the string to ANS_STRING.
: 1749      3254 1 | --
: 1750      3255 1 |
: 1751      3256 2 | BEGIN
: 1752      3257 2 |
: 1753      3258 2 | LOCAL
: 1754      3259 2 |     BUF_DESC      : BLOCK [8, BYTE] VOLATILE, | Local temporary buffer
: 1755      3260 2 |     BUF           : REF VECTOR [65535, BYTE], | Addresses temp buffer BUF_DESC
: 1756      3261 2 |     PUTTER        : INITIAL (0), | Counter for BUF
: 1757      3262 2 |     STRING_BUF    : REF VECTOR [65535, BYTE], | Addresses STRING
: 1758      3263 2 |     STRING_LEN    : WORD, | Length of STRING
: 1759      3264 2 |     HIGH_POS      : | Max power of ten in result
: 1760      3265 2 |     LOW_POS       : | Min power of ten in result
: 1761      3266 2 |     SIGN_STR      : BYTE INITIAL (BYTE(0)), | Temp to hold sign of string
: 1762      3267 2 |     FIRST_GOOD    : INITIAL (0), | Used for leading zero suppression,
: 1763      3268 2 | | signals first non-zero significant
: 1764      3269 2 | | digit was encountered.
: 1765      3270 2 |     ZERO_OK       : INITIAL (0), | Used for leading zero suppression,
: 1766      3271 2 | | signals that the only 'ok' leading
: 1767      3272 2 | | zero is the zero in front of the
: 1768      3273 2 | | decimal point if the # is a fraction.
: 1769      3274 2 |     DOT_HERE      : INITIAL (0), |
: 1770      3275 2 |     EXTRA         : WORD INITIAL (0), | # of extra leading zeroes in a
: 1771      3276 2 | | COMP data item that resulted
: 1772      3277 2 | | from the conversion process.
: 1773      3278 2 |     PULL          : WORD INITIAL (0), | Keeps track of the # of leading
: 1774      3279 2 | | zeroes pulled from STRING.
: 1775      3280 2 |     MINUS         : INITIAL (0); | Signals that a minus sign is
: 1776      3281 2 | | to be inserted
: 1777      3282 2 |
: 1778      3283 2 | BIND
: 1779      3284 2 |     ZERO = UPLIT ('0') ;
: 1780      3285 2 |
: 1781      3286 2 | | +
: 1782      3287 2 | | Enable a handler to free the local string in case of error.
: 1783      3288 2 | | -
: 1784      3289 2 |
: 1785      3290 2 | ENABLE
: 1786      3291 2 |     COB$$$FREE_STRINGS (BUF_DESC);
```



```
1787 3292 2
1788 3293 2
1789 3294 2
1790 3295 2
1791 3296 2
1792 3297 2
1793 3298 2
1794 3299 2
1795 3300 2
1796 3301 2
1797 3302 2
1798 3303 2
1799 3304 2
1800 3305 2
1801 3306 2
1802 3307 2
1803 3308 2
1804 3309 2
1805 3310 2
1806 3311 2
1807 3312 2
1808 3313 2
1809 3314 2
1810 3315 2
1811 3316 2
1812 3317 2
1813 3318 2
1814 3319 2
1815 3320 2
1816 3321 2
1817 3322 2
1818 3323 2
1819 3324 2
1820 3325 2
1821 3326 2
1822 3327 2
1823 3328 2
1824 3329 2
1825 3330 2
1826 3331 2
1827 3332 2
1828 3333 2
1829 3334 2
1830 3335 2
1831 3336 2
1832 3337 2
1833 3338 2
1834 3339 2
1835 3340 2
1836 3341 2
1837 3342 2
1838 3343 2
1839 3344 2
1840 3345 2
1841 3346 2
1842 3347 2
1843 3348 2

      + Create local descriptor BUF_DESC. Allocate enough space to hold the
      | digits, a leading sign (or space), and an imbedded decimal point (or
      | comma).
      | Calculate limits for loop that reads digits.
      -

STRING_LEN = .STRING [DSC$W_LENGTH];
STRING_BUF = .STRING [DSC$A_POINTER];

BUF_DESC [DSC$W_LENGTH] = 0;
BUF_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
BUF_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
BUF_DESC [DSC$A_POINTER] = 0;

      + Calculate limits for loop that reads digits.
      |
      -

HIGH_POS = MAX (.STRING_LEN + .EXPONENT - 1, -1);
LOW_POS = MIN (.EXPONENT, 0);

      +
      | If the resultant number has too many digits to be represented on
      | VAX, give an error message.
      -

IF ((.HIGH_POS - .LOW_POS + 3) GTR 65535)
THEN
  LIB$STOP (COB$_INVARG) ;

      + Allocate space for local string.
      | STR$COPY_R allocates space for ANS_STRING but not for BUF_DESC.
      -

IF NOT ( STR$GET1_DX (%REF (.HIGH_POS - .LOW_POS + 3), BUF_DESC) )
THEN
  LIB$STOP (COB$ERRDURDIS) ;
BUF = .BUF_DESC [DSC$A_POINTER];

      + Calculate number of extra leading zeroes introduced in DISP_CONVERT
      | that were not part of the original input string passed to COB$DISPLAY.
      |
      -

IF .DIGITS EQL -1
THEN
  EXTRA = 5 ;
IF .DIGITS GTR 0
THEN
  EXTRA = .STRING_LEN - .DIGITS ;

      + Read sign, put either '+', '-', or space into BUF, incr PUTTER
      | (Can't read trailing sign yet)
```

```
1844 3349 2 !-
1845 3350 2
1846 3351 2 IF .LOOK_FOR_SIGN EQL 1 OR .LOOK_FOR_SIGN EQL 3
1847 3352 2 THEN
1848 3353 2
1849 3354 2
1850 3355 2
1851 3356 2
1852 3357 2
1853 3358 2
1854 3359 2
1855 3360 2
1856 3361 2
1857 3362 2
1858 3363 2
1859 3364 2
1860 3365 2
1861 3366 2
1862 3367 2
1863 3368 2
1864 3369 2
1865 3370 2
1866 3371 2
1867 3372 2
1868 3373 2
1869 3374 2
1870 3375 2
1871 3376 2
1872 3377 2
1873 3378 2
1874 3379 2
1875 3380 2
1876 3381 2
1877 3382 2
1878 3383 2
1879 3384 2
1880 3385 2
1881 3386 2
1882 3387 2
1883 3388 2
1884 3389 2
1885 3390 2
1886 3391 2
1887 3392 2
1888 3393 2
1889 3394 2
1890 3395 2
1891 3396 2
1892 3397 2
1893 3398 2
1894 3399 2
1895 3400 2
1896 3401 2
1897 3402 2
1898 3403 2
1899 3404 2
1900 3405 2

IF .LOOK_FOR_SIGN EQL 1 OR .LOOK_FOR_SIGN EQL 3
THEN
    +
    These data types will not output a '+' if the number was positive,
    a space will be output instead.
    -
    BEGIN
    LOCAL
    P_IND : INITIAL (0); ! Equals 1 if string has P in picture
    IF .STRING [DSC$B_CLASS] EQL DSC$K_CLASS_SD
    THEN
        IF (P_IND = ABS(.STRING [DSC$B_SCALE]) GTR .STRING [DSC$B_DIGITS])
        THEN
            +
            Strings with Ps in the picture have the sign in STRING_BUF[0].
            Replace sign with zero so there won't be a double sign in
            the result.
            -
            BEGIN
            SIGN_STR = .STRING_BUF [0] ;
            CH$MOVE (1, ZERO, STRING_BUF [0]);
            END
        ELSE
            SIGN_STR = .STRING_BUF [(EXPO + .STRING_LEN - 1) - .HIGH_POS]
    ELSE
        SIGN_STR = .STRING_BUF [(EXPO + .STRING_LEN - 1) - .HIGH_POS] ;

    +
    If no sign in original input string, insert a space before
    the number.
    Do not output the plus sign if present - output a
    space instead. Minus sign will be inserted IMMEDIATELY before the
    first significant digit ( bb-12.3 not -bb12.3 ). For now,
    put a space in BUF as a place holder, take care of sign
    insertion later.
    -
    BUF [.PUTTER] = %C' ' ;
    PUTTER = .PUTTER + 1 ;

    +
    A minus sign is always included for output. Signal through
    MINUS = 1 that a minus sign is to be inserted in BUF.
    -
    IF .SIGN_STR EQL %C'+' OR .SIGN_STR EQL %C'-'
    THEN
        BEGIN
        IF .SIGN_STR EQL %C'-'
        THEN
```



```

: 1901      3406      4      MINUS = 1 ;
: 1902      3407      5      IF (.LOOK_FOR_SIGN EQL 1) AND (NOT (.P_IND))
: 1903      3408      4      THEN
: 1904      3409      4          HIGH_POS = .HIGH_POS - 1 ;
: 1905      3410      4
: 1906      3411      3      END ;
: 1907      3412      3
: 1908      3413      2      END;
: 1909      3414      2
: 1910      3415      2
: 1911      3416      2      + Case where minus sign was lost in routine DISP_CONVERT - signal through
: 1912      3417      2      MINUS = 1 that a minus sign is to be inserted in BUF. Minus sign will
: 1913      3418      2      be inserted IMMEDIATELY before the first significant digit ( bb-12.3
: 1914      3419      2      not -bb12.3 ). For now, put a space in BUF as a place holder, take
: 1915      3420      2      care of sign insertion later, incr PUTTER.
: 1916      3421      2      -
: 1917      3422      2      IF .LOOK_FOR_SIGN EQL 4
: 1918      3423      2      THEN
: 1919      3424      2          BEGIN
: 1920      3425      2
: 1921      3426      2          MINUS = 1 ;
: 1922      3427      2          SIGN_STR = 'C'-' ' ;
: 1923      3428      2          BUF [PUTTER] = 'C' ' ;
: 1924      3429      2          PUTTER = .PUTTER + 1 ;
: 1925      3430      2
: 1926      3431      2          END ;
: 1927      3432      2
: 1928      3433      2      +
: 1929      3434      2      Must create a dummy string for strings with picture of the form
: 1930      3435      2      9(x)P(x). The reason for this is that what is in STRING does not
: 1931      3436      2      reflect the placeholders at all and therefore the code that puts
: 1932      3437      2      the result in BUF does not work properly.
: 1933      3438      2      NOTE: NEW_DIGITS
: 1934      3439      2      -
: 1935      3440      2      IF (.STRING [DSC$B_CLASS] EQL DSC$K_CLASS_SD) AND (.DIGITS EQL 0)
: 1936      3441      2      THEN IF .STRING [DSC$B_SCALE] GTR 0
: 1937      3442      2      THEN
: 1938      3443      2          BEGIN
: 1939      3444      2              +
: 1940      3445      2              Picture is of the form 9(x)P(x).
: 1941      3446      2              -
: 1942      3447      2              LOCAL
: 1943      3448      2                  DUMMY : BLOCK [8,BYTE], ! Dummy string which will have placeholders in it
: 1944      3449      2                  DUM_STR : VECTOR [20,BYTE], ! Must hold up to 18 numeric string
: 1945      3450      2                  ! digits - also making room for sign
: 1946      3451      2                  ! and decimal point
: 1947      3452      2
: 1948      3453      2                  NUM_CHARS;
: 1949      3454      2
: 1950      3455      2                  DUMMY [DSC$W_LENGTH] = 20;
: 1951      3456      2                  DUMMY [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 1952      3457      2                  DUMMY [DSC$B_CLASS] = DSC$K_CLASS_S;
: 1953      3458      2                  DUMMY [DSC$A_POINTER] = DUM_STR [0];
: 1954      3459      2
: 1955      3460      2              +
: 1956      3461      2              Zero the whole dummy string so that zeroes will end up wherever
: 1957      3462      2              the digits and sign aren't
: 1957      3462      2              -

```

```
1958 3463 3 NUM CHARS = .DUMMY [DSC$W_LENGTH];
1959 3464 3 STR$DUPL_CHAR (DUMMY, NUM_CHARS, ZERO);
1960 3465 3
1961 3466 3 IF .STRING [DSC$B_DTYPE] EQL DSC$K_DTYPE_NR
1962 3467 3 THEN
1963 3468 3 BEGIN
1964 3469 3 +
1965 3470 3 | Right separate -
1966 3471 3 | Dummy string should have the digits in STRING moved
1967 3472 3 | into it first, then the placeholder zeroes, then the
1968 3473 3 | sign.
1969 3474 3 -
1970 3475 3 NUM CHARS = .STRING [DSC$W_LENGTH] - 1; ! Number of digits
1971 3476 3 CH$MOVE (.NUM_CHARS, .STRING [DSC$A_POINTER], .DUMMY [DSC$A_POINTER]);
1972 3477 3 CH$MOVE (1, .STRING [DSC$A_POINTER] + .NUM_CHARS, .DUMMY [DSC$A_POINTER] + .NUM_CHARS + .EXPONEN
1973 3478 3
1974 3479 3 END
1975 3480 3 ELSE
1976 3481 3 IF .STRING [DSC$W_LENGTH] EQL .STRING [DSC$B_DIGITS]
1977 3482 3 THEN
1978 3483 3 +
1979 3484 3 | Left and right overpunched -
1980 3485 3 | Dummy string should have the sign moved into it first,
1981 3486 3 | then the digits in STRING, then the placeholder zeroes.
1982 3487 3 -
1983 3488 3 CH$MOVE (.STRING [DSC$W_LENGTH] + 1, .STRING [DSC$A_POINTER], .DUMMY [DSC$A_POINTER])
1984 3489 3
1985 3490 3 ELSE
1986 3491 3 +
1987 3492 3 | Left separate -
1988 3493 3 | Dummy string should have the digits in STRING moved
1989 3494 3 | into it first then the placeholder zeroes.
1990 3495 3 -
1991 3496 3 BEGIN
1992 3497 3 CH$MOVE (.STRING [DSC$W_LENGTH] - 1, .STRING [DSC$A_POINTER] + 1, .DUMMY [DSC$A_POINTER]);
1993 3498 3 HIGH_POS = .HIGH_POS - 1;
1994 3499 3 STRING_LEN = .STRING_LEN - 1;
1995 3500 3
1996 3501 3 END;
1997 3502 3
1998 3503 3 STRING_BUF = .DUMMY [DSC$A_POINTER];
1999 3504 3
2000 3505 3 END;
2001 3506 3
2002 3507 3 +
2003 3508 3 | Now read (rest of) number inserting decimal point (or comma).
2004 3509 3 | Put result in BUF.
2005 3510 3 -
2006 3511 3
2007 3512 3 DECR POS FROM .HIGH_POS TO .LOW_POS DO
2008 3513 3 BEGIN
2009 3514 3 ! Begin loop
2010 3515 3 IF (.POS EQL -1)
2011 3516 3 THEN
2012 3517 3 BEGIN
2013 3518 3 ! Decimal point/comma insertion
2014 3519 3 | When pos = -1 we are about to
2015 3520 3 | read the first digit to the
2016 3521 3 | right of the decimal point
```



```
2015 3520 4      | Do not suppress zeroes immediately following the decimal point
2016 3521 4      | (.002 should not get .2 as a result)
2017 3522 4      | If the decimal point is the first significant character in the
2018 3523 4      | number, check to see if it is necessary to insert a minus sign
2019 3524 4      | before the decimal point. ( .002 might be -.002 )
2020 3525 4      |
2021 3526 4      |
2022 3527 4      | IF .HIGH_POS EQL -1
2023 3528 4      | THEN
2024 3529 5      | BEGIN
2025 3530 5      | FIRST_GOOD = 1 ;
2026 3531 5      | IF .MINUS EQL 1
2027 3532 5      | THEN
2028 3533 5      |     BUF [.PUTTER - 1] = .SIGN_STR ;
2029 3534 4      | END ;
2030 3535 4      |
2031 3536 4      |
2032 3537 4      | + When requested, use a comma in place of a decimal point,
2033 3538 4      | -
2034 3539 4      |
2035 3540 4      | IF (.FLAGS AND V_DEC_PT) NEQ 0
2036 3541 4      | THEN
2037 3542 4      |     BUF [.PUTTER] = %C','
2038 3543 4      | ELSE
2039 3544 4      |     BUF [.PUTTER] = %C'.' ;
2040 3545 4      | DOT_HERE = .PUTTER ;
2041 3546 4      | PUTTER = .PUTTER + 1 ;
2042 3547 4      | END ;
2043 3548 4      |
2044 3549 4      |
2045 3550 4      | + Read number, one digit at a time. Put digit in BUF, incr PUTTER
2046 3551 4      | -
2047 3552 4      |
2048 3553 4      | IF .LOOK_FOR_SIGN EQL 2 AND .POS EQL .LOW_POS
2049 3554 4      | THEN
2050 3555 4      |
2051 3556 4      |
2052 3557 4      | + Trailing sign - this case also outputs a space instead of a
2053 3558 4      | + sign.
2054 3559 4      | When .POS = .LOW_POS we are reading the last digit, if we
2055 3560 4      | have a trailing sign data type - that last digit is the sign.
2056 3561 4      | Make sure there is something in BUF before inserting the sign.
2057 3562 4      | -
2058 3563 4      |
2059 3564 4      | BEGIN
2060 3565 4      |
2061 3566 4      | IF .PUTTER EQL 0
2062 3567 4      | THEN
2063 3568 5      | BEGIN
2064 3569 5      |     BUF [.PUTTER] = %C'0' ;
2065 3570 5      |     PUTTER = 1 ;
2066 3571 5      | END
2067 3572 4      | ELSE
2068 3573 4      |     IF .BUF [.PUTTER - 1] EQL %C' '
2069 3574 4      |     THEN
2070 3575 4      |         BUF [.PUTTER - 1] = %C'0' ;
2071 3576 4      |
```

```
! Check for nothing in
! BUF
```

```
! Check for only spaces
! in BUF
```

```
2072 3577 4 SIGN_STR = .STRING_BUF [( .EXPONENT + .STRING_LEN - 1 ) - .POS] ;
2073 3578 4 IF .SIGN_STR EQL %C'+
2074 3579 4 THEN
2075 3580 4 BUF [.PUTTER] = %C' '
2076 3581 4 ELSE
2077 3582 4 BUF [.PUTTER] = .SIGN_STR ;
2078 3583 4 END
2079 3584 3 ELSE
2080 3585 4 BEGIN
2081 3586 4 +
2082 3587 4 ??? This needs a comment.
2083 3588 4 -
2084 3589 5 IF ((.POS GTR (.STRING_LEN + .EXPONENT - 1)) OR (.POS LSS .EXPONENT))
2085 3590 4 THEN
2086 3591 4 +
2087 3592 4 Put trailing, but significant, zeroes in BUF.
2088 3593 4 (zeroes to left of decimal point)
2089 3594 4 NOTE - this also puts placeholder zeroes to the right of
2090 3595 4 the decimal point for STRING with picture of form P(x)9(x).
2091 3596 4 This is why a dummy string did not have to be set up for
2092 3597 4 that form.
2093 3598 4 -
2094 3599 4 BUF [.PUTTER] = %C'0'
2095 3600 4 ELSE
2096 3601 4
2097 3602 4 +
2098 3603 4 Put digit in BUF.
2099 3604 4 -
2100 3605 4
2101 3606 4 IF .LOOK_FOR_SIGN GEQ 0 AND .LOOK_FOR_SIGN LEQ 2
2102 3607 4 THEN
2103 3608 5 BUF [.PUTTER] = .STRING_BUF [( .EXPONENT + .STRING_LEN - 1 )
2104 3609 4 - .POS]
2105 3610 4 ELSE
2106 3611 5 BUF [.PUTTER] = .STRING_BUF [( .EXPONENT + .STRING_LEN - 1 )
2107 3612 4 - (.POS - 1)] ;
2108 3613 3 END ;
2109 3614 3
2110 3615 3 +
2111 3616 3 Search for leading zeroes.
2112 3617 3 If a leading zero is encountered - replace it with a space,
2113 3618 3 and leave FIRST_GOOD set at 0.
2114 3619 3 The first digit encountered that is not a zero will turn off
2115 3620 3 the search by setting FIRST_GOOD to 1.
2116 3621 3 -
2117 3622 3
2118 3623 3 IF .FIRST_GOOD EQL 0
2119 3624 3 THEN
2120 3625 3 IF .BUF [.PUTTER] EQL %C'0'
2121 3626 3 THEN
2122 3627 4 IF (.ZERO_OK EQL 0 AND
2123 3628 5 ((.POS EQL 0) OR
2124 3629 7 ((.POS EQL 1) AND (.STRING[DSC$B_DTYPE] EQL DSC$K_DTYPE_NR)
2125 3630 6 AND (.EXPONENT GEQ 0))
2126 3631 4 ) AND .YES_ZERO EQL 1)
2127 3632 3 THEN
2128 3633 4 BEGIN
```



```
2129 3634 4
2130 3635 4
2131 3636 4
2132 3637 4
2133 3638 4
2134 3639 4
2135 3640 4
2136 3641 4
2137 3642 5
2138 3643 5
2139 3644 5
2140 3645 5
2141 3646 5
2142 3647 5
2143 3648 5
2144 3649 5
2145 3650 4
2146 3651 4
2147 3652 5
2148 3653 5
2149 3654 5
2150 3655 5
2151 3656 5
2152 3657 5
2153 3658 5
2154 3659 5
2155 3660 5
2156 3661 4
2157 3662 4
2158 3663 4
2159 3664 4
2160 3665 5
2161 3666 5
2162 3667 5
2163 3668 5
2164 3669 5
2165 3670 5
2166 3671 5
2167 3672 5
2168 3673 5
2169 3674 5
2170 3675 5
2171 3676 5
2172 3677 5
2173 3678 5
2174 3679 5
2175 3680 5
2176 3681 5
2177 3682 5
2178 3683 5
2179 3684 5
2180 3685 5
2181 3686 5
2182 3687 5
2183 3688 5
2184 3689 5
2185 3690 5

      - Leave zero before decimal point if number is less
      - than 1. ( ie. -.1 should be output as -0.1 )
      ZERO OK = 1 ;
      FIRST GOOD = 1 ;
      IF .MINUS EQL 1
      THEN
        BEGIN
          + Insert minus sign immediately before the first
          + significant digit. Reset MINUS to zero to signal
          + completion of sign insertion.
          -
          BUF [.PUTTER - 1] = .SIGN_STR ;
          MINUS = 0 ;
          END ;
      ELSE
        IF .DIGITS NEQ 0
        THEN
          + Pull out leading zeroes that were introduced in the
          + conversion process (to COMP data items).
          -
          IF .PULL LSS .EXTRA
          THEN
            BEGIN
              PUTTER = .PUTTER - 1 ;
              PULL = .PULL + 1 ;
              END
            ELSE
              + No more 'extra' leading zeroes to pull. Replace
              + leading zeroes that were part of the original
              + string (string passed to COB$DISPLAY) with
              + spaces.
              -
              if .putter lss .dot_here or .dot_here eql 0
              then
                BUF [.PUTTER] = %C' '
              else
                buf [.putter] = %c'0'
            ELSE
              + We are not dealing with a COMP data item (original
              + string before conversion), there are no 'extra'
              + leading zeroes. Replace a leading zero, that was
              + part of the original string, with a space.
              -
              if .putter lss .dot_here or .dot_here eql 0
              then
                BUF [.PUTTER] = %C' '
              else
                buf [.putter] = %c'0'
            ELSE
      ELSE
```

```
2186 3691 4 BEGIN
2187 3692 4 +
2188 3693 4 | We have encountered a digit that is not a zero, stop search
2189 3694 4 | for leading zeroes.
2190 3695 4 -
2191 3696 4 FIRST_GOOD = 1 ;
2192 3697 4 IF .MINUS EQL 1
2193 3698 4 THEN
2194 3699 4 +
2195 3700 4 | Insert minus sign immediately before the first
2196 3701 4 | significant digit. No need to reset MINUS as
2197 3702 4 | FIRST_GOOD = 1 prevents this section of code from
2198 3703 4 | being entered again.
2199 3704 4 -
2200 3705 4 | Do not write over the decimal point if we have
2201 3706 4 | space - dec pt - significant digit in BUF
2202 3707 4 | Bug was comp SV9 value -.9 giving -9
2203 3708 4 -
2204 3709 4 IF .DOT_HERE EQL 1
2205 3710 4 THEN
2206 3711 4 | BUF [.PUTTER - 2] = .SIGN_STR
2207 3712 4 ELSE
2208 3713 4 | BUF [.PUTTER - 1] = .SIGN_STR ;
2209 3714 3 END ;
2210 3715 3
2211 3716 3 PUTTER = .PUTTER + 1;
2212 3717 3
2213 3718 3 END;
2214 3719 2 ! End loop
2215 3720 2
2216 3721 2 +
2217 3722 2 | Check that at least one zero digit is DISPLAYed if the number is zero.
2218 3723 2 | Do not allow only a sign, decimal point/comma or space to be output
2219 3724 2 | without at least one zero.
2220 3725 2 | Getting " . " want " .0"
2221 3726 2 -
2222 3727 2
2223 3728 4 IF ((.PUTTER EQL 3) AND (.BUF[0] EQL %C' ' OR .BUF[0] EQL %C'-' )
2224 3729 4 | AND (.BUF[1] EQL %C'.' OR .BUF[1] EQL %C' ,')
2225 3730 4 | AND (.BUF[2] EQL %C' ' OR .BUF[2] EQL %C'-' ))
2226 3731 3 THEN
2227 3732 3 BEGIN ! space dot space
2228 3733 3 BUF[2] = %C'0' ;
2229 3734 2 END ;
2230 3735 2
2231 3736 2 +
2232 3737 2 | Avoid display of too many digits for a COMP item - look at the
2233 3738 2 | DIGITS and SCALE field of the input STRING (from ACC_CONVERT)
2234 3739 2 | Trunc/Notrunc issue
2235 3740 2 -
2236 3741 2
2237 3742 2 IF .CHECK_COMP EQL 1
2238 3743 2 THEN
2239 3744 3 BEGIN ! Begin COMP data item check
2240 3745 3 LOCAL ! # of digits you SHOULD have
2241 3746 3 LEFT_DEC ! to left and right of dec pt
2242 3747 3 RIGHT_DEC,
```



```
2243 3748 3          HAVE_LEFT          ! # of digits you DO have
2244 3749 3          HAVE_RIGHT ;        ! to left and right of dec pt
2245 3750 3
2246 3751 3          IF .COMP_SCALE NEQ 0
2247 3752 3          THEN
2248 3753 4              BEGIN
2249 3754 4                  LEFT_DEC = .DIGITS + .COMP_SCALE ; ! Class SD - look at DIGITS
2250 3755 4                  RIGHT_DEC = .DIGITS - .LEFT_DEC ; ! field for 'true' number of
2251 3756 4                  END ! digits to DISPLAY
2252 3757 3          ELSE
2253 3758 4              BEGIN
2254 3759 4                  LEFT_DEC = .DIGITS ; ! Class S - VAX COBOL has passed
2255 3760 4                  RIGHT_DEC = 0 ; ! an SD descriptor - look at
2256 3761 4                  END ; ! DIGITS field for 'true' number
2257 3762 4                  ! of digits to DISPLAY
2258 3763 4
2259 3764 4          !+ Is what I have within the limits ?
2260 3765 4          ! If what we have is greater than what we expect (according to DIGITS),
2261 3766 4          ! pull the extra digits.
2262 3767 4          ! ex: Pic 99v99 gets 123.456 but should be changed to 23.45
2263 3768 4          !-
2264 3769 4
2265 3770 3          IF .DOT_HERE NEQ 0
2266 3771 3          THEN
2267 3772 4              BEGIN ! Begin SD Class
2268 3773 4          !+
2269 3774 4              ! Use DOT_HERE (postion of decimal point in BUF) to calculate
2270 3775 4              ! the character you have to the left and right of the dec pt.
2271 3776 4          !-
2272 3777 4              HAVE_LEFT = .DOT_HERE - 1 ; ! -1 for sign
2273 3778 4              HAVE_RIGHT = .PUTTER - .DOT_HERE - 1 ;
2274 3779 4
2275 3780 5              IF (.HAVE_LEFT GTR .LEFT_DEC OR
2276 3781 5                  .HAVE_RIGHT GTR .RIGHT_DEC )
2277 3782 4              THEN ! Too many digits
2278 3783 5                  BEGIN
2279 3784 5                      LOCAL
2280 3785 5                      TEMP : VECTOR [25, BYTE],
2281 3786 5                      Y : INITIAL (0), ! Ptr to TEMP
2282 3787 5                      K : INITIAL (0) ; ! New PUTTER
2283 3788 5
2284 3789 5                      INCR X FROM 1 TO .DOT_HERE - 1 DO ! Peel from BUF backwards
2285 3790 5                      ! Leave sign alone
2286 3791 6                      BEGIN
2287 3792 6                          TEMP [.Y] = .BUF [.DOT_HERE - .X] ;
2288 3793 6                          Y = .Y + 1 ;
2289 3794 6                      END ;
2290 3795 5
2291 3796 5                      DECR I FROM .LEFT_DEC TO 1 DO ! Put desired # of
2292 3797 6                      ! digits before dec pt.
2293 3798 6                      BEGIN
2294 3799 6                          BUF [.I] = .TEMP [.K] ;
2295 3800 6                          K = .K + 1 ;
2296 3801 6                      END ;
2297 3802 5                      BUF [.K+1] = .BUF [.DOT_HERE] ; ! Put dec pt/comma in
2298 3803 5                      K = .K + 2 ; ! new position in BUF
2299 3804 5
```



```
2300      3805 5      INCR S FROM 1 TO .RIGHT_DEC DO      ! Put desired # of
2301      3806 6      BEGIN                                ! digits after dec pt.
2302      3807 6      BUF [.K] = .BUF [.DOT_HERE + .S] ;
2303      3808 6      K = .K + 1 ;
2304      3809 5      END ;
2305      3810 5      PUTTER = .K ;
2306      3811 5      END ;
2307      3812 4      END ;                                ! End SD Class
2308      3813 4      ELSE
2309      3814 3      BEGIN                                ! Begin S Class
2310      3815 4      +
2311      3816 4      PUTTER = 1 (for sign) is the number of characters you have.
2312      3817 4      -
2313      3818 4      HAVE_LEFT = .PUTTER - 1 ;
2314      3819 4      HAVE_RIGHT = 0 ;
2315      3820 4      IF (.HAVE_LEFT GTR .LEFT_DEC OR
2316      3821 4      .HAVE_RIGHT GTR .RIGHT_DEC )
2317      3822 5      THEN
2318      3823 5      BEGIN
2319      3824 4      LOCAL
2320      3825 5      TEMP : VECTOR [25, BYTE],
2321      3826 5      Y : INITIAL (0),
2322      3827 5      K : INITIAL (0) ;
2323      3828 5      ! Ptr to TEMP
2324      3829 5      ! New PUTTER
2325      3830 5      INCR X FROM 1 TO .PUTTER - 1 DO
2326      3831 5      ! Peel from BUF backwards
2327      3832 5      ! Leave sign alone
2328      3833 6      BEGIN
2329      3834 6      TEMP [.Y] = .BUF [.PUTTER - .X] ;
2330      3835 6      Y = .Y + 1 ;
2331      3836 5      END ;
2332      3837 5      DECR I FROM .LEFT_DEC TO 1 DO
2333      3838 5      ! Put desired # of
2334      3839 6      BEGIN                                ! digits before dec pt.
2335      3840 6      BUF [.I] = .TEMP [.K] ;
2336      3841 6      K = .K + 1 ;
2337      3842 5      END ;
2338      3843 5      PUTTER = .K + 1 ;
2339      3844 4      END ;
2340      3845 3      END ;                                ! End Class S
2341      3846 2      END ;                                ! End COMP data item check
2342      3847 2
2343      3848 2
2344      3849 2      +
2345      3850 2      Distinguish between signed COMP/COMP3 and unsigned COMP/COMP3.
2346      3851 2      Look at bit 7 of FLAGS passed by the COBOL Compiler. There is
2347      3852 2      no such data item in the COBOL compiler as an unsigned COMP.
2348      3853 2      TO get rid of sign - shift contents of BUF up 1, overwriting the sign
2349      3854 2      with is in BUF [0]. Decrement PUTTER by 1.
2350      3855 2      -
2351      3856 2      IF ( .FLAGS AND V_NO_SIGN ) NEQ 0 AND .CHECK_COMP NEQ 0
2352      3857 2      THEN
2353      3858 2      BEGIN
2354      3859 2      +
2355      3860 2      There is a difference between a string coming into the DISPLAY
2356      3861 2      routines from the ACCEPT routines than one that does not
```



```
2357 3862 1      ! pass through the ACCEPT routines.
2358 3863 1      !-
2359 3864 1      LOCAL
2360 3865 1      Y      :  INITIAL (0) ;
2361 3866 1
2362 3867 3      INCR X FROM 1 TO .PUTTER - 1 DO
2363 3868 4      BEGIN
2364 3869 4      BUF [.Y] = .BUF [.X] ;
2365 3870 4      Y = .Y + 1 ;
2366 3871 3      END ;
2367 3872 3      PUTTER = .PUTTER - 1 ;
2368 3873 3      END ;
2369 3874 2
2370 3875 2      !+
2371 3876 2      Copy the final form of the string to ANS_STRING
2372 3877 2      !-
2373 3878 2
2374 3879 3      IF NOT ( STR$COPY_R (.ANS_STRING, PUTTER, .BUF ) )
2375 3880 2      THEN
2376 3881 2      LIB$STOP (COB$_ERRDURDIS) ;
2377 3882 2
2378 3883 2      !+
2379 3884 2      Free local string.
2380 3885 2      No need to worry about problems with VM as the length of BUF_DESC has
2381 3886 2      not been changed since STR$GET1_DX (BUF_DESC).
2382 3887 2      !-
2383 3888 2
2384 3889 2      STR$FREE1_DX (BUF_DESC);
2385 3890 1      END;                                     ! end of DISP_PARSE
```

```
00 00 00 30 009B1 .BLKB 3
009B4 P.AAT: .ASCII \0\<0><0><0>
ZERO= P.AAT
```

```
OFFC 00000 DISP_PARSE:
SE AC AE 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 3178
2C AE D4 00006 MOVAB -84(SP), SP
20 AE 94 00009 CLRL PUTTER : 3256
18 AE 7C 0000C CLRB SIGN_STR
57 D4 0000F CLRL ZERO_OK
14 AE B4 00011 CLRL DOT_HERE
10 AE B4 00014 CLRL EXTRA
0C AE D4 00017 CLRL PULL
4C AE 7C 0001A CLRL MINUS
6D 044F CF DE 0001D MOVAL BUF_DESC
5A 04 AC D0 00022 MOVL 72$, (FP)
08 AE 6A B0 00026 MOVW STRING, R10 : 3300
04 AE AA D0 0002A MOVW (R10), STRING_LEN
59 04 AE D0 0002F MOVL 4(R10), 4(SP) : 3301
4E AE 4C AE B4 00033 MOVL 4(SP), STRING_BUF
4F AE 0E 90 00036 CLRW BUF_DESC : 3303
02 90 0003A MOVB #14, BUF_DESC+2 : 3304
MOVB #2, BUF_DESC+3 : 3305
```

		50	AE	D4	0003E	CLRL	BUF_DESC+4	3306
		08	AE	3C	00041	MOVZWL	STRING_LEN, R2	3312
		10	AC	D0	00045	MOVL	EXPONENT, R11	
		50	FF	AB42	9E	MOVAB	-1(R11)[R2], R0	
	FFFFFFF	8F		50	D1	CMPL	R0, #-1	
				03	18	BGEQ	1\$	
		50		01	CE	MNEGL	#1, R0	
		58		50	D0	MOVL	R0, HIGH_POS	
		50		5B	D0	MOVL	R11, R0	3313
				02	15	BLEQ	2\$	
				50	D4	CLRL	R0	
		6E		50	D0	MOVL	R0, LOW_POS	
50		58		6E	C3	SUBL3	LOW_POS, HIGH_POS, R0	3320
		53	03	A0	9E	MOVAB	3(R0), R3	
	0000FFFF	8F		53	D1	CMPL	R3, #65535	
				0D	15	BLEQ	3\$	
			00000000G	8F	DD	PUSHL	#COB\$ INVARG	3322
	00000000G	00		01	FB	CALLS	#1, LIB\$STOP	
			4C	AE	9F	PUSHAB	BUF_DESC	3329
	28	AE		53	D0	MOVL	R3, -40(SP)	
			28	AE	9F	PUSHAB	40(SP)	
	00000000G	00		02	FB	CALLS	#2, STR\$GET1_DX	
		0D		50	E8	BLBS	R0, 4\$	
			00000000G	8F	DD	PUSHL	#COB\$ ERRDURDIS	3331
	00000000G	00		01	FB	CALLS	#1, LIB\$STOP	
		56	50	AE	D0	MOVL	BUF_DESC+4, BUF	3332
	FFFFFFF	8F	14	AC	D1	CMPL	DIGITS, #-1	3339
				04	12	BNEQ	5\$	
	14	AE		05	B0	MOVW	#5, EXTRA	3341
			14	AC	D5	TSTL	DIGITS	3342
				06	15	BLEQ	6\$	
14	AE	52	14	AC	A3	SUBW3	DIGITS, R2, EXTRA	3344
				54	D4	CLRL	R4	3351
		01	0C	AC	D1	CMPL	LOOK_FOR_SIGN, #1	
				04	12	BNEQ	7\$	
				54	D6	INCL	R4	
				06	11	BRB	8\$	
		03	0C	AC	D1	CMPL	LOOK_FOR_SIGN, #3	
				61	12	BNEQ	15\$	
				53	D4	CLRL	P_IND	3359
		09	03	AA	91	CMPB	3(R10), #9	3364
				26	12	BNEQ	11\$	
		51	08	AA	98	CVTBL	8(R10), R1	3366
				03	18	BGEQ	9\$	
		51		51	CE	MNEGL	R1, R1	
				50	D4	CLRL	R0	
51	09	AA	08	00	ED	CMPZV	#0, #8, 9(R10), R1	
				02	18	BGEQ	10\$	
				50	D6	INCL	R0	
		53		50	D0	MOVL	R0, P_IND	
		0B		50	E9	BLBC	R0, 1T\$	
	20	AE		69	90	MOVB	(STRING_BUF), SIGN_STR	3375
		69	FEFC	CF	90	MOVB	ZERO, (STRING_BUF)	3376
				0D	11	BRB	12\$	3366
		50		52	C1	ADDL3	R2, R11, R0	3382
		50		58	C2	SUBL2	HIGH_POS, R0	
	20	AE	FF	A049	90	MOVB	-1(R0)[STRING_BUF], SIGN_STR	

2C	BE46	2C	20	90	00110	12\$:	MOVB	#32, @PUTTER[BUF]	3393
		2C	AE	D6	00115		INCL	PUTTER	3394
	2B	20	AE	91	00118		CMPB	SIGN_STR, #43	3400
			06	13	0011C		BEQL	13\$	
	2D	20	AE	91	0011E		CMPB	SIGN_STR, #45	
			12	12	00122		BNEQ	15\$	
	2D	20	AE	91	00124	13\$:	CMPB	SIGN_STR, #45	3404
			04	12	00128		BNEQ	14\$	
0C	AE		01	D0	0012A		MOVL	#1, MINUS	3406
	05		54	E9	0012E	14\$:	BLBC	R4, 15\$	3407
	02		53	E8	00131		BLBS	P_IND, 15\$	
			58	D7	00134		DECL	HIGH_POS	3409
	04	0C	AC	D1	00136	15\$:	CMPL	LOOK_FOR_SIGN, #4	3422
			10	12	0013A		BNEQ	16\$	
0C	AE		01	D0	0013C		MOVL	#1, MINUS	3426
20	AE		2D	90	00140		MOVB	#45, SIGN_STR	3427
2C	BE46		20	90	00144		MOVB	#32, @PUTTER[BUF]	3428
		2C	AE	D6	00149		INCL	PUTTER	3429
	09	03	AA	91	0014C	16\$:	CMPB	3(R10), #9	3440
			03	12	00150		BNEQ	17\$	
		14	AC	D5	00152		TSTL	DIGITS	
			7C	12	00155	17\$:	BNEQ	21\$	
		08	AA	95	00157		TSTB	8(R10)	3441
			77	15	0015A		BLEQ	21\$	
44	AE	010E0014	8F	D0	0015C		MOVL	#17694740, DUMMY	3454
48	AE		30	AE	9E	00164	MOVAB	DUM_STR, DUMMY+4	3457
28	AE		44	AE	3C	00169	MOVZWL	DUMMY, NUM_CHARS	3463
		FE8A	CF	9F	0016E		PUSHAB	ZERO	3464
		2C	AE	9F	00172		PUSHAB	NUM_CHARS	
		4C	AE	9F	00175		PUSHAB	DUMMY	
00000000G	00		03	FB	00178		CALLS	#3, STR\$DUPL_CHAR	
	12	02	AA	91	0017F		CMPB	2(R10), #18	3466
			20	12	00183		BNEQ	18\$	
28	AE		6A	3C	00185		MOVZWL	(R10), NUM_CHARS	3475
		28	AE	D7	00189		DECL	NUM_CHARS	
48	BE	04	BE	28	0018C		MOV3	NUM_CHARS, @4(SP), @DUMMY+4	3476
	50	48	AE	28	AE	C1	ADDL3	NUM_CHARS, DUMMY+4, R0	3477
		51	AE	04	D0	00199	MOVL	4(SP), R1	
	6B40	28	BE41	90	0019D		MOVB	@NUM_CHARS[R1], (R11)[R0]	
			2A	11	001A3		BRB	20\$	3466
	50	09	AA	9A	001A5	18\$:	MOVZBL	9(R10), R0	3481
	6A		50	B1	001A9		CMPW	R0, (R10)	
			0D	12	001AC		BNEQ	19\$	
	50		6A	3C	001AE		MOVZWL	(R10), R0	3488
			50	D6	001B1		INCL	R0	
48	BE	04	BE	50	28	001B3	MOV3	R0, @4(SP), @DUMMY+4	
		50		14	11	001B9	BRB	20\$	
			6A	3C	001BB	19\$:	MOVZWL	(R10), R0	3497
	50		50	D7	001BE		DECL	R0	
		04	AE	01	C1	001C0	ADDL3	#1, 4(SP), -(SP)	
48	7E		9E	50	28	001C5	MOV3	R0, @4(SP)+, @DUMMY+4	
	BE			58	D7	001CA	DECL	HIGH_POS	3498
				AE	B7	001CC	DECW	STRING_LEN	3499
	59	08	AE	D0	001CF	20\$:	MOVL	DUMMY+4, STRING_BUF	3503
	52	08	AE	3C	001D3	21\$:	MOVZWL	STRING_LEN, R2	3577
	50		58	D0	001D7		MOVL	HIGH_POS, POS	
			015C	31	001DA		BRW	44\$	

	FFFFFFF	8F		50	D1	001DD	22\$:	CMPL	POS, #-1		3515
	FFFFFFF	8F		36	12	001E4		BNEQ	26\$		
				58	D1	001E6		CMPL	HIGH_POS, #-1		3527
				14	12	001ED		BNEQ	23\$		
	1C	AE		01	D0	001EF		MOVL	#1, FIRST_GOOD		3530
		01	0C	AE	D1	001F3		CMPL	MINUS, #1		3531
				0A	12	001F7		BNEQ	23\$		
51		56	2C	AE	C1	001F9		ADDL3	PUTTER, BUF, R1		3533
	FF	A1	20	AE	90	001FE		MOVB	SIGN_STR, -1(R1)		
53		56	2C	AE	C1	00203	23\$:	ADDL3	PUTTER, BUF, R3		3542
05	08	AC		06	E1	00208		BBC	#6, FLAGS, 24\$		3540
		63		2C	90	0020D		MOVB	#4, (R3)		3542
				03	11	00210		BRB	25\$		
		63		2E	90	00212	24\$:	MOVB	#46, (R3)		3544
		57	2C	AE	D0	00215	25\$:	MOVL	PUTTER, DOT_HERE		3545
			2C	AE	D6	00219		INCL	PUTTER		3546
		51	2C	AE	D0	0021C	26\$:	MOVL	PUTTER, R1		3566
		02	0C	AC	D1	00220		CMPL	LOOK_FOR_SIGN, #2		3553
				42	12	00224		BNEQ	30\$		
		6E		50	D1	00226		CMPL	POS, LOW_POS		
				3D	12	00229		BNEQ	30\$		
				51	D5	0022B		TSTL	R1		3566
				0A	12	0022D		BNEQ	27\$		
		6146		30	90	0022F		MOVB	#48, (R1)[BUF]		3569
	2C	AE		01	D0	00233		MOVL	#1, PUTTER		3570
				0C	11	00237		BRB	28\$		3566
		20	FF	A146	91	00239	27\$:	CMPB	-1(R1)[BUF], #32		3573
				05	12	0023E		BNEQ	28\$		
51	FF	A146		30	90	00240		MOVB	#48, -1(R1)[BUF]		3575
		5B		52	C1	00245	28\$:	ADDL3	R2, R11, R1		3577
		51		50	C2	00249		SUBL2	POS, R1		
	20	AE	FF	A149	90	0024C		MOVB	-1(R1)[STRING_BUF], SIGN_STR		
53		56	2C	AE	C1	00252		ADDL3	PUTTER, BUF, R3		3580
		2B	20	AE	91	00257		CMPB	SIGN_STR, #43		3578
				05	12	0025B		BNEQ	29\$		
		63		20	90	0025D		MOVB	#32, (R3)		3580
				40	11	00260		BRB	34\$		
		63	20	AE	90	00262	29\$:	MOVB	SIGN_STR, (R3)		3582
				3A	11	00266		BRB	34\$		3553
53		56		51	C1	00268	30\$:	ADDL3	R1, BUF, R3		3599
		51	FF	AB42	9E	0026C		MOVAB	-1(R11)[R2], R1		3589
		51		50	D1	00271		CMPL	POS, R1		
				05	14	00274		BGTR	31\$		
		5B		50	D1	00276		CMPL	POS, R11		
				05	18	00279		BGEQ	32\$		
		63		30	90	0027B	31\$:	MOVB	#48, (R3)		3599
				22	11	0027E		BRB	34\$		
		51	FF	A24B	9E	00280	32\$:	MOVAB	-1(R2)[R11], R1		3608
			0C	AC	D5	00285		TSTL	LOOK_FOR_SIGN		3606
				10	19	00288		BLSS	33\$		
		02	0C	AC	D1	0028A		CMPL	LOOK_FOR_SIGN, #2		
				0A	14	0028E		BGTR	33\$		
54		51		50	C3	00290		SUBL3	POS, R1, R4		3609
		63		6449	90	00294		MOVB	(R4)[STRING_BUF], (R3)		3608
				08	11	00298		BRB	34\$		
		51		50	C2	0029A	33\$:	SUBL2	POS, R1		3612
		63	01	A149	90	0029D		MOVB	1(R1)[STRING_BUF], (R3)		3611

		1C	AE	D5	002A2	34\$:	TSTL	FIRST_GOOD	3623	
			70	12	002A5		BNEQ	41\$		
	30		63	91	002A7		CMPB	(R3), #48	3625	
			63	12	002AA		BNEQ	40\$		
		18	AE	D5	002AC		TSTL	ZERO_OK	3627	
			36	12	002AF		BNEQ	36\$		
			50	D5	002B1		TSTL	POS	3628	
			0F	13	002B3		BEQL	35\$		
	01		50	D1	002B5		CMPL	POS, #1	3629	
			2D	12	002B8		BNEQ	36\$		
	12	02	AA	91	002BA		CMPB	2(R10), #18		
			27	12	002BE		BNEQ	36\$		
			5B	D5	002C0		TSTL	R11	3630	
			23	19	002C2		BLSS	36\$		
	01	1C	AC	D1	002C4	35\$:	CMPL	YES_ZERO, #1	3631	
			1D	12	002C8		BNEQ	36\$		
	18	AE	01	D0	002CA		MOVL	#1, ZERO_OK	3638	
	1C	AE	01	D0	002CE		MOVL	#1, FIRST_GOOD	3639	
		01	OC	AE	D1	002D2	CMPL	MINUS, #1	3640	
			5C	12	002D6		BNEQ	43\$		
51			2C	AE	C1	002D8	ADDL3	PUTTER, BUF, R1	3648	
	FF	56	20	AE	90	002DD	MOVB	SIGN_STR, -1(R1)		
		A1	OC	AE	D4	002E2	CLRL	MINUS	3649	
			4D	11	002E5		BRB	43\$	3627	
			14	AC	D5	002E7	36\$:	TSTL	DIGITS	3653
			0F	13	002EA		BEQL	37\$		
	14	AE	10	AE	B1	002EC	CMPW	PULL, EXTRA	3659	
			08	1E	002F1		BGEQU	37\$		
			2C	AE	D7	002F3	DECL	PUTTER	3662	
			10	AE	B6	002F6	INCL	PULL	3663	
			39	11	002F9		BRB	43\$	3659	
	57		2C	AE	D1	002FB	37\$:	CMPL	PUTTER, DOT_HERE	3684
			04	19	002FF		BLSS	38\$		
			57	D5	00301		TSTL	DOT_HERE		
			05	12	00303		BNEQ	39\$		
	63		20	90	00305	38\$:	MOVB	#32, (R3)	3686	
			2A	11	00308		BRB	43\$		
	63		30	90	0030A	39\$:	MOVB	#48, (R3)	3688	
			25	11	0030D		BRB	43\$	3627	
	1C	AE	01	D0	0030F	40\$:	MOVL	#1, FIRST_GOOD	3696	
		01	OC	AE	D1	00313	CMPL	MINUS, #1	3697	
			1B	12	00317	41\$:	BNEQ	43\$		
		01	57	D1	00319		CMPL	DOT_HERE, #1	3709	
			OC	12	0031C		BNEQ	42\$		
51		56	2C	AE	C1	0031E	ADDL3	PUTTER, BUF, R1	3711	
	FE	A1	20	AE	90	00323	MOVB	SIGN_STR, -2(R1)		
			0A	11	00328		BRB	43\$		
51		56	2C	AE	C1	0032A	42\$:	ADDL3	PUTTER, BUF, R1	3713
	FF	A1	20	AE	90	0032F	MOVB	SIGN_STR, -1(R1)		
			2C	AE	D6	00334	43\$:	INCL	PUTTER	3717
			50	D7	00337		DECL	POS	3512	
	6E		50	D1	00339	44\$:	CMPL	POS, LOW_POS		
			03	19	0033C		BLSS	45\$		
			FE9C	31	0033E		BRW	22\$		
	03		2C	AE	D1	00341	45\$:	CMPL	PUTTER, #3	3728
			26	12	00345		BNEQ	49\$		
	20		66	91	00347		CMPB	(BUF), #32		

			05	13	0034A	BEQL	46\$		
	2D		66	91	0034C	CMPB	(BUF), #45		
			1C	12	0034F	BNEQ	49\$		
	2E	01	A6	91	00351	46\$: CMPB	1(BUF), #46		3729
			06	13	00355	BEQL	47\$		
	2C	01	A6	91	00357	CMPB	1(BUF), #44		
			10	12	0035B	BNEQ	49\$		
	20	02	A6	91	0035D	47\$: CMPB	2(BUF), #32		3730
			06	13	00361	BEQL	48\$		
	2D	02	A6	91	00363	CMPB	2(BUF), #45		
			04	12	00367	BNEQ	49\$		
	02	A6	30	90	00369	48\$: MOVB	#48, 2(BUF)		3733
	01		24	AC	D1	49\$: CMPL	CHECK_COMP, #1		3742
			77	12	00371	BNEQ	59\$		
			20	AC	D5	00373	TSTL	COMP_SCALE	
			0D	13	00376	BEQL	50\$		3751
51			20	AC	C1	00378	ADDL3	COMP_SCALE, DIGITS, LEFT_DEC	3754
59	14	AC	51	C3	0037E	SUBL3	LEFT_DEC, DIGITS, RIGHT_DEC		3755
			06	11	00383	BRB	51\$		3751
	51		14	AC	D0	00385	50\$: MOVL	DIGITS, LEFT_DEC	3759
			59	D4	00389	CLRL	RIGHT_DEC		3760
			57	D5	0038B	51\$: TSTL	DOT_HERE		3770
			5D	13	0038D	BEQL	60\$		
	55		FF	A7	9E	0038F	MOVAB	-1(R7), R5	3777
	52		55	D0	00393	MOVL	R5, HAVE_LEFT		
50	2C	AE	57	C3	00396	SUBL3	DOT_HERE, PUTTER, R0		3778
			50	D7	0039B	DECL	HAVE_RIGHT		
	51		52	D1	0039D	CMPL	HAVE_LEFT, LEFT_DEC		3780
			05	14	003A0	BGTR	52\$		
	59		50	D1	003A2	CMPL	HAVE_RIGHT, RIGHT_DEC		3781
			57	15	003A5	BLEQ	61\$		
			50	D4	003A7	52\$: CLRL	K		3783
			52	7C	003A9	CLRL	Y		
			0C	11	003AB	BRB	54\$		3790
54		57	53	C3	003AD	53\$: SUBL3	X, DOT_HERE, R4		3792
	30	AE42	6446	90	003B1	MOVB	(R4)[BUF], TEMP[Y]		
			52	D6	003B7	INCL	Y		3793
F0		53	55	F3	003B9	54\$: AOBLEQ	R5, X, 53\$		3790
			51	D6	003BD	INCL	I		3796
			08	11	003BF	BRB	56\$		
	6146		30	AE40	90	003C1	55\$: MOVB	TEMP[K], (I)[BUF]	3798
			50	D6	003C7	INCL	K		3799
	F5		51	F5	003C9	56\$: SOBGTR	I, 55\$		3796
	01	A046	6746	90	003CC	MOVB	(DOT_HERE)[BUF], 1(K)[BUF]		3802
		50	02	C0	003D2	ADDL2	#2, R		3803
			51	D4	003D5	CLRL	S		3805
			09	11	003D7	BRB	58\$		
52		57	51	C1	003D9	57\$: ADDL3	S, DOT_HERE, R2		3807
		8046	6246	90	003DD	MOVB	(R2)[BUF], (K)+[BUF]		
F3		51	59	F3	003E2	58\$: AOBLEQ	RIGHT_DEC, S, 57\$		3805
	2C	AE	50	D0	003E6	MOVL	K, PUTTER		3810
			3F	11	003EA	59\$: BRB	67\$		3770
55	2C	AE	01	C3	003EC	60\$: SUBL3	#1, PUTTER, R5		3819
		52	55	D0	003F1	MOVL	R5, HAVE_LEFT		
			50	D4	003F4	CLRL	HAVE_RIGHT		3820
		51	52	D1	003F6	CMPL	HAVE_LEFT, LEFT_DEC		3822
			05	14	003F9	BGTR	62\$		

59		50	D1	003FB		CMPL	HAVE_RIGHT, RIGHT_DEC	3823		
		2B	15	003FE	61\$:	BLEQ	67\$	3825		
		50	D4	00400	62\$:	CLRL	Y	3832		
		52	7C	00402		CLRQ	K	3834		
		0D	11	00404		BRB	64\$	3835		
54	2C	AE	53	C3	00406	63\$:	SUBL3	X, PUTTER, R4	3836	
	30	AE40	6446	90	0040B		MOVB	(R4)[BUF], TEMP[Y]	3837	
			50	D6	00411		INCL	Y	3838	
EF		53	55	F3	00413	64\$:	AOBLEQ	R5, X, 63\$	3839	
			51	D6	00417		INCL	I	3840	
			08	11	00419		BRB	66\$	3841	
		6146	30	AE42	90	0041B	65\$:	MOVB	TEMP[K], (I)[BUF]	3842
			52	D6	00421		INCL	K	3843	
			51	F5	00423	66\$:	SOBGR	I, 65\$	3844	
	2C	F5	01	A2	9E	00426	MOVAB	1(R2), PUTTER	3845	
		AE	08	AC	95	0042B	67\$:	TSTB	FLAGS	3846
			16	18	0042E		BGEQ	70\$		
			24	AC	D5	00430	TSTL	CHECK_COMP		
			11	13	00433		BEQL	70\$		
			50	7C	00435		CLRQ	X	3867	
			05	11	00437		BRB	69\$		
		8146	6046	90	00439	68\$:	MOVB	(X)[BUF], (Y)+[BUF]	3869	
F6		50	2C	AE	F2	0043E	69\$:	AOBLSS	PUTTER, X, 68\$	3870
			2C	AE	D7	00443		DECL	PUTTER	3871
				56	DD	00446	70\$:	PUSHL	BUF	3872
			30	AE	9F	00448		PUSHAB	PUTTER	3873
			18	AC	DD	0044B		PUSHL	ANS_STRING	
	00000000G	00		03	FB	0044E		CALLS	#3, STR\$COPY_R	
		0D		50	E8	00455		BLBS	R0, 71\$	
			00000000G	8F	DD	00458		PUSHL	#COB\$ERRDURDIS	3881
	00000000G	00		01	FB	0045E		CALLS	#1, LIB\$STOP	
			4C	AE	9F	00465	71\$:	PUSHAB	BUF_DESC	3889
	00000000G	00		01	FB	00468		CALLS	#1, STR\$FREE1_DX	
				04	0046F		RET			3890
				0000	00470	72\$:	.WORD	Save nothing		3256
		50	08	AC	D0	00472	MOVL	8(AP), R0		
		50	04	A0	D0	00476	MOVL	4(R0), R0		
			F8	A0	9F	0047A	PUSHAB	BUF_DESC		
				01	DD	0047D	PUSHL	#1		
				5E	DD	0047F	PUSHL	SP		
		7E	04	AC	7D	00481	MOVQ	4(AP), -(SP)		
	0000V	CF		03	FB	00485	CALLS	#3, COB\$\$FREE_STRINGS		
				04	0048A		RET			

; Routine Size: 1163 bytes, Routine Base: _COB\$CODE + 09B8

```
2387 3891 1 GLOBAL ROUTINE COB$$FREE_STRINGS (
2388 3892 1     SIG,
2389 3893 1     MECH,
2390 3894 1     ENBL
2391 3895 1 ) =
2392 3896 1
2393 3897 1 ++
2394 3898 1 FUNCTIONAL DESCRIPTION:
2395 3899 1
2396 3900 1     If we are unwinding, free the local strings. They are passed
2397 3901 1     in the enable vector.
2398 3902 1
2399 3903 1 FORMAL PARAMETERS:
2400 3904 1
2401 3905 1     SIG.rl.a      A counted vector of parameters to LIB$SIGNAL/STOP
2402 3906 1     MECH.rl.a     A counted vector of info from CHF
2403 3907 1     ENBL.ra.a     A counted vector of ENABLE argument addresses.
2404 3908 1
2405 3909 1 IMPLICIT INPUTS:
2406 3910 1
2407 3911 1     NONE
2408 3912 1
2409 3913 1 IMPLICIT OUTPUTS:
2410 3914 1
2411 3915 1     NONE
2412 3916 1
2413 3917 1 ROUTINE VALUE:
2414 3918 1 COMPLETION CODES:
2415 3919 1
2416 3920 1     Always SSS_RESIGNAL, which is ignored when unwinding.
2417 3921 1
2418 3922 1 SIDE EFFECTS:
2419 3923 1
2420 3924 1     Frees all of the strings passed as enable arguments.
2421 3925 1
2422 3926 1 --
2423 3927 1
2424 3928 2 BEGIN
2425 3929 2
2426 3930 2 MAP
2427 3931 2     SIG : REF VECTOR,
2428 3932 2     MECH : REF VECTOR,
2429 3933 2     ENBL : REF VECTOR;
2430 3934 2
2431 3935 2 ++
2432 3936 2 Only free the strings if this is the UNWIND condition.
2433 3937 2 --
2434 3938 2
2435 3939 2 ! IF ( NOT (LIB$MATCH_COND (SIG [1], %REF (SS$_UNWIND))) ) THEN RETURN (SS$_RESIGNAL);
2436 3940 2
2437 3941 2 ++
2438 3942 2 Go through the enable arguments, freeing them.
2439 3943 2 --
2440 3944 2
2441 3945 2 INCR ARG_NO FROM 1 TO .ENBL [0] DO
2442 3946 2
2443 3947 2     IF (..ENBL [.ARG_NO] NEQ 0) THEN STR$FREE1_DX (.ENBL [.ARG_NO]);
```


COB\$DISPLAY
1-015

VAX-11 COBOL DISPLAY statement

L 8
16-Sep-1984 00:02:31
14-Sep-1984 12:10:42

VAX-11 Bliss-32 V4.0-742
[COBRTL.SRC]COBDISPLA.B32;1

Page 69
(13)

: 2444
: 2445
: 2446

3948 2
3949 2
3950 1

RETURN (SS\$_RESIGNAL);
END;

! end of COB\$\$FREE_STRINGS

0004 00000
52 D4 00002
12 11 00004
50 0C BC42 D0 00006 1\$:
60 D5 0000B
09 13 0000D
50 DD 0000F
00000000G 00
E9 52 0C BC F3 00018 2\$:
50 0918 8F 3C 0001D
04 00022

.ENTRY COB\$\$FREE_STRINGS, Save R2
CLRL ARG_NO
BRB 2\$
MOVL @ENBL[ARG_NO], R0
TSTL (R0)
BEQL 2\$
PUSHL R0
CALLS #1, STR\$FREE1_DX
AOBLEQ @ENBL, ARG_NO, 1\$
MOVZWL #2328, R0
RET

: 3891
: 3945
: 3947
: 3949
: 3950

; Routine Size: 35 bytes, Routine Base: _COB\$CODE + 0E43

```
: 2448      3951 1 GLOBAL ROUTINE COB$$RET_A_AB_PREV =
: 2449      3952 1
: 2450      3953 1 !++
: 2451      3954 1 FUNCTIONAL DESCRIPTION:
: 2452      3955 1
: 2453      3956 1     Returns address of COB$$AB_PREV for code outside the COBRTL image
: 2454      3957 1     that needs this variable.
: 2455      3958 1
: 2456      3959 1 FORMAL PARAMETERS:
: 2457      3960 1
: 2458      3961 1     NONE
: 2459      3962 1
: 2460      3963 1 IMPLICIT INPUTS:
: 2461      3964 1
: 2462      3965 1     NONE
: 2463      3966 1
: 2464      3967 1 IMPLICIT OUTPUTS:
: 2465      3968 1
: 2466      3969 1     NONE
: 2467      3970 1
: 2468      3971 1 ROUTINE VALUE:
: 2469      3972 1 COMPLETION CODES:
: 2470      3973 1
: 2471      3974 1     Address of COB$$AB_PREV.
: 2472      3975 1
: 2473      3976 1 SIDE EFFECTS:
: 2474      3977 1
: 2475      3978 1     NONE
: 2476      3979 1
: 2477      3980 1 !--
: 2478      3981 1
: 2479      3982 2 BEGIN
: 2480      3983 2
: 2481      3984 2 RETURN COB$$AB_PREV;
: 2482      3985 2
: 2483      3986 1 END;                                     ! end of COB$$RET_A_AB_PREV
```

```
50 00000000' EF 0000 00000
                      9E 00002
                      04 00009
```

```
.ENTRY COB$$RET_A_AB_PREV, Save nothing
MOVAB COB$$AB_PREV, -R0
RET
```

```
: 3951
: 3984
: 3986
```

; Routine Size: 10 bytes, Routine Base: _COB\$CODE + 0E66

```
: 2484      3987 1
: 2485      3988 1 END
: 2486      3989 0 ELUDOM
```

! end of module COB\$DISPLAY

COB\$DISPLAY
1-015

VAX-11 COBOL DISPLAY statement

N 8
16-Sep-1984 00:02:31
14-Sep-1984 12:10:42

VAX-11 Bliss-32 V4.0-742
[COBRTL.SRC]COBDISPLA.B32;1

Page 71
(14)

PSECT SUMMARY

Name	Bytes	Attributes
COB\$DATA	55	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
COB\$CODE	3696	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	120	1	581	00:00.8
\$255\$DUA28:[COBRTL.OBJ]SMGLIB.L32;1	469	4	0	38	00:00.3

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:COBDISPLA/OBJ=OBJ\$:COBDISPLA MSRC\$:COBDISPLA/UPDATE=(ENH\$:COBDISPLA)

: Size: 3443 code + 308 data bytes
: Run Time: 00:51.7
: Elapsed Time: 03:19.7
: Lines/CPU Min: 4625
: Lexemes/CPU-Min: 29863
: Memory Used: 476 pages
: Compilation Complete

0062 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

